DOCUMENT RESUME

ED 063 007 LI 003 689

AUTHOR	Murray, Daniel McClure
TITLE	Document Retrieval Based on Clustered Files.
INSTITUTION	Cornell Univ., Ithaca, N.Y. Dept. of Computer Science.
SPONS AGENCY	National Library of Medicine (DHEW), Bethesda, Md.; National Science Foundation, Washington, D.C.
REPORT NO	ISR-20
PUB DATE	May 72
NOTE	357p.; 97 References; A Thesis Presented to the Faculty of the Graduate School of Cornell University for the Degree of Doctor of Philosophy
EDRS PRICE	MF-$0.65 HC-$13.16
DESCRIPTORS	*Cluster Grouping; Doctoral Theses; *Documentation; *Information Retrieval; *Information Storage; Information Systems

ABSTRACT

A retrieval system is considered in which document descriptions are stored and accessed in groups called clusters. All items in a cluster meet common similarity criteria and are represented by a composite entity called a profile. In large collections, profiles themselves are clustered and additional levels of profiles are generated. This entire process establishes a file organization for the system in that records are composed with a logical structure with a directory (profile hierarchy) to facilitate searching. Clustered files have the following advantages over other organizations: complete document information is stored in the same location; storage overhead is low; and flexible, economical searches can be realized. The problems investigated in clustered file organization are: profile definition, updating, hierarchy storage, and secondary profile uses. A comparison with an inverted file is included. Nearly all work has an experimental base and uses the SMART retrieval system. The proposed organization compares favorably in terms of speed and storage economy. Various request-document matching procedures, and feedback schemes are easily implemented. Search precision is less, but compensated by a flexible level of recall--low or high. (Author/SJ)

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEH REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ED 063007

Department of Computer Science .
Cornell University
Ithaca, New York 14850

FILMED FROM BEST AVAILABLE COPY

Scientific Report No. ISR-20

INFORMATION STORAGE AND RETRIEVAL

to

The National Science Foundation

and to

The National Library of Medicine

Document Retrieval Based on Clustered Files

Ithaca, New York
May 1972

Gerard Salton
Project Director

LI 003 689

©

Copyright, 1972
by Cornell University

Use, reproduction, or publication, in whole or in part, is permitted for any purpose of the United States Government.

DOCUMENT RETRIEVAL BASED ON CLUSTERED FILES

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University for the Degree of

Doctor of Philosophy

by

Daniel McClure Murray

June, 1972

The author, Daniel McClure Murray, was born in Anderson, Indiana on May 28, 1944. After attending public schools, he entered Purdue University and subsequently received his Bachelor of Science in mathematics in 1966. At Cornell University, he earned his Master of Science (1968) and Doctor of Philosophy (1972) degrees in computer science, the latter degree with specialization in information retrieval. While at Cornell, he published articles in the Journal of Library Automation (September, 1970) and in the Information Storage and Retrieval (ISR) series of the Computer Science Department. Since October 1970, the author has been employed in Research and Engineering Division of the Xerox Corporation in Rochester, New York. He is a member of the Association for Computing Machinery. In 1968, he married the former Pamela Richeson of Anderson, Indiana.

4

To my wife Pam

whose support and encouragement

made it possible to complete this thesis

Acknowledgments

The author would like to thank Professor G. Salton for chairing his Special Committee, directing this thesis, and giving time and advice as needed. This association has been most rewarding. Professors A. C. Shaw and W. Maxwell are acknowledged as the other members of the Special Committee; Professor J. Williams helped with the final thesis review. The guidance of Professors D. M. Jackson and K. Needham was most helpful for the work on term-substitutes (Chapter VIII). The entire staff of the SMART Retrieval System is commended for providing a test vehicle for some of these experiments; of particular note are Messrs. Zumoff, Worona, Crawford, and Williamson. Thanks to Mr. R. T. Dattola for the use of his clustering program.

The National Science Foundation is recognized for the support of the author's education including summer programs at Indiana University (1961) and the University of California San Diego (1962) and support for graduate work at Cornell University (1966-1970).

A special acknowledgment is given to the author's wife for her never ending patience, financial and moral support, and preparation of the thesis draft.

Table of Contents

Abstract

Synopsis

Chapter I. Introduction

1. Automated Information Systems I-1
   - A. General System Types I-1
   - B. Document Retrieval Systems I-3
2. File Organizations I-5
   - A. General Purpose and Definition I-5
   - B. File Organization in a Document Retrieval System I-7
3. The Document Retrieval Environment I-9
   - A. Characteristics of a Document File I-9
   - B. User Factors I-11
   - C. Software Services I-12
   - D. Storage Devices I-16
4. Clustered File Organizations I-19
   - A. Overall Concepts I-19
   - B. Areas for Investigation I-21
5. Outline I-22

Chapter II. Survey of File Organizations

1. Introduction II-1
2. Methods of Logical Organization II-2
   - A. Sequential Files II-2

B. Chained Files II-5

C. Inverted Files II-10

D. Calculated Access Files II-18

E. Clustered Files II-23

3. Methods of Physical Organization II-26

4. Summary II-30

Chapter III. Clustered Files

1. Introduction III-1

2. Classification Methods III-1

3. Hierarchy Formation III-5

A. General Structure III-6

B. Linkage Among Nodes III-7

C. Profile Definition III-11

4. Search Strategies III-18

5. Updating III-21

6. Hierarchy Storage III-27

A. General III-27

B. A Disk Storage Algorithm III-28

C. Orders for Hierarchy Storage III-30

7. Query Clustering III-31

8. Alternate Uses of the Hierarchy III-32

A. Suggested Uses III-33

B. Query Alteration III-35

9. Summary III-39

3

Chapter IV. The Experimental Environment

1. Introduction to the SMART System IV-1

2. The Data Collections IV-4

3. The Generated Hierarchies IV-8

4. Evaluation IV-11

    A. SMART Evaluation IV-12

    B. Cluster-Oriented Evaluation IV-20

Chapter V. Profile Experiments

1. Introduction V-1

2. Standard Profile Performance V-2

3. Rank Value Profiles V-7

    A. Base Value Selection V-8

    B. Weight Origins and Apexes V-13

    C. Weight Range V-16

    D. Summary V-22

4. Search Bias V-24

    A. An Algorithm for Detecting Bias V-24

    B. Investigation of Biased Searches V-31

5. Profile Length V-42

6. Frequency Considerations V-53

7. Unweighted and Partially Weighted Profiles V-61

8. Summary of Results for Hierarchy 1 V-73

9. Confirmation Tests V-76

10. Discussion V-82

Chapter VI. File Maintenance Experiments

1. Introduction VI-1
2. Method VI-3
3. Profile Maintenance Procedures VI-10
4. Degeneration of the Hierarchy VI-19
5. Summary VI-30

Chapter VII. Experiments with Hierarchy Storage

1. Introduction VII-1
2. Procedure VII-2
3. Test Results VII-10
4. Summary VII-16

Chapter VIII. Experiments with a Query Alteration Scheme
Based on a Cluster Hierarchy

1. Introduction VIII-1
2. Deriving Base-Substitute Pairs VIII-2
3. Term Substitutes as Precision and Recall Devices VIII-10
4. Summary VIII-18

Chapter IX. Comparison of Inverted and Clustered Document Files

1. Introduction IX-1
2. The Inverted Directory--Storage and Search IX-3
3. Comparison of Inverted and Clustered Document Files IX-11
4. Summary IX-18

Chapter X. Summary, Conclusions, Discussion, and Suggestions for Future Work X-1

Appendix A. Common Word List A-1

Appendix B. Subcollections for the Updating Experiments B-1

Appendix C. Confirmation Test Evaluation Curves C-1

List of Tables

Chapter I

I-1 Storage Capacity and Retrieval Time for the IBM 2314 Direct Access Storage Facility

Chapter II

II-1 Evaluation Summary for Logical Access Methods

Chapter IV

IV-1 Properties of the Experimental Hierarchies
IV-2 SMART Search Parameters for Cluster Searches

Chapter V

V-1 Properties of P_1, P_2, P_3 Profiles for Hierarchy 1
V-2 Profile Term Deletion Strategies
V-3 Profile Length Reduction Resulting from Term Deletion Strategy 3, Hierarchy 1, P_3^* Profiles
V-4 Summary of Hierarchy Properties
V-5 Relative Merit of Selected Profiles in Confirmation Tests

Chapter VI

VI-1 Properties of the Original Clustered Collections Before Updating
VI-2 Properties of the Updated Collections

Chapter VII

VII-1 Parameters Related to Management of Disk Storage
VII-2 Hierarchy Storage in Level and Heir-filial Order

VII-3 Average Expansion Characteristics of SMART Searches, Hierarchy 1, $P^*_3(\delta = -1)$ Profiles
VII-4 Average Expansion Characteristics of Simulated Searches Using the Cranfield Query Set
VII-5 Average Expansion Characteristics of Simulated Searches Using Random Selection of Expanded Nodes (Simulated Queries)
VII-6,7 I/O Activity in Simulated Cluster Searches
VII-8 Relation of Performance and I/O Activity for Cluster and Full Searches
Chapter IX
IX-1 Parameters for the Management of the Inverted Document File
IX-2 Results of Storing the Inverted Directory
IX-3 Comparison of I/O Requirements in Inverted and Clustered File Searches
Appendix C
C-1 Notation for the Confirmation Tests

List of Illustrations

Chapter I

I-1 Typical Search Negotiation in an On-Line Document Retrieval System

I-2 Sample Hierarchy Resulting from Document Clustering

Chapter II

II-1 Structure and Search of a Document File Using a Chained Organization

II-2 Structure and Search of a Document File Using Lists of Accession Numbers Inverted by Index Term

II-3 Sample Correlation Calculation Using an Inverted File with Contributions Stored in the Directory Lists

II-4 File Organization Based on Query and Document Code Words

II-5 Indexed Sequential Access

Chapter III

III-1 Storage of a Cluster Hierarchy

III-2 Profiles Resulting from Unweighted Document Vectors

III-3 Profiles Resulting from Weighted Document Vectors

III-4 Rank Value Weighting Applied to Standard Profiles

III-5 Updating a Clustered File

III-6 Use of Term Substitutes in Cluster Searching

Chapter IV

IV-1 Model Document Vector

IV-2 Properties of the Cranfield Document and Query Collection
IV-3 Distribution of Relevant Documents
IV-4 Example of Precision-Recall Evaluation
IV-5 Sample Precision-Recall Curve for Hierarchy 1
IV-6 Example of Cluster-Oriented Evaluation
IV-7 Sample Precision Floor-Recall Ceiling Curves
IV-8 Best Achievable Performance Curves

Chapter V

V-1,2 Evaluation of the Standard Profile Definitions, Hierarchy 1
V-3,4 Evaluation of Search Performance as a Function of Base Value, Rank Value P_2 Profiles, Hierarchy 1
V-5,6 Comparison of Fixed and Variable Weight Origins, Rank Value P_2 Profiles, Hierarchy 1
V-7,8 Comparison of Profile Term Weights Based on Frequency Counts and Frequency Ranks, Hierarchy 1
V-9 Cosine Correlation Contribution Ratios, Hierarchy 1
V-10 Construction of P_2^* and P_3^* Profiles
V-11 Examples of the Analysis of Search Results for Bias
V-12,13, 14,15 Behavioral Characteristics of Profiles, Rank Value Weights Based on Document Frequencies (P_2), Hierarchy 1
V-16,17 18,19 Behavioral Characteristics of Standard and Modified Profiles, Hierarchy 1

V-20 Distribution of Profile Terms by Frequency

V-21,22 Search Performance After Deletion of Profile Terms, P^*_3 Profiles, Hierarchy 1

V-23 Performance Loss Due to Deletion of High Weight Profile Terms, Hierarchy 1

V-24 Luhn's Hypothetical Relationship Between Significance and Term Frequency

V-25,26 Contribution Ratios Resulting From Bending, Hierarchy 1

V-27,28 Search Performance Resulting From Profiles with Increasing-Decreasing Contribution Curves, Hierarchy 1

V-29 Performance of Unweighted Profiles Using a Size Dependent Cosine Function, Hierarchy 1

V-30 Behavioral Characteristics of Unweighted Profiles Using a Size Dependent Cosine Function, Hierarchy 1

V-31,32 Performance of Unweighted Profiles with Term Deletion, Hierarchy 1

V-33,34 Behavioral Characteristics of Unweighted Profiles with Term Deletion, Hierarchy 1

V-35 Performance of Profiles with Full, Partial, and No Weights, Hierarchy 1, $P^*_3(\delta = -1)$ Profiles

Chapter VI

VI-1,2, 3,4,5 Comparison of Profile Maintenance Procedures, Hierarchies 4,5,6

VI-6,7,8, 9,10,11 Hierarchy Degeneration Resulting From Updating

Chapter VIII

VIII-1 Computation of a Term-Term Association Matrix

VIII-2,3 Flowchart for Deriving Base-Substitute Pairs

VIII-4 Relative Merit of Using Substitutes as Precision Devices, Hierarchy 1

VIII-5,6 Relative Merit of Using Substitutes as Recall Devices, Hierarchy 1

Chapter IX

IX-1 Inverted Directory I/O (Disk Accesses) as a Function of Query Length

IX-2 Comparison of Precision-Recall Data From Inverted and Clustered File Searches

Appendix C

C-1,2,3,4 Confirmation Tests, Hierarchy 1

C-5,6,7,8 Confirmation Tests, Hierarchy 2

C-9,10,11, 12 Confirmation Tests, Hierarchy 3

DOCUMENT RETRIEVAL BASED ON CLUSTERED FILE

Daniel McClure Murray, Ph.D.

Cornell University, 1972

A retrieval system is considered in which document descriptions are stored and accessed in groups called clusters. All items in a cluster meet common similarity criteria and are represented by a composite entity called a profile. In large collections, profiles themselves are clustered and additional levels of profiles are generated. This entire process establishes a file organization for the system in that records are composed into a logical structure with a directory (profile hierarchy) to facilitate searching. Clustered files have the following advantages over other organizations: complete document information is stored in the same location, storage overhead is low, and flexible and economical searches can be realized.

The problems investigated in clustered file organizations are: profile definition, updating, hierarchy storage, and secondary profile uses. A comparison with an inverted file is included also. Nearly all work has an experimental base and uses the SMART retrieval system or facilities built around it. In this report, the initial chapters cover concepts in document retrieval, file organization, and clustered files. Chapter IV describes the experimental environment and a new evaluation scheme for cluster searches based on precision floor and recall ceiling. Chapter V deals with the preparation of unbiased, economical profiles. Several types (standard, rank value, rank,

shortened) and weighting schemes (none, partial, full) are studied. A reasonable profile can be constructed by using term weights based on frequency ranks and deleting a large percentage of low weight terms. Chapter VI indicates that profiles require only minor weight adjustments to incorporate new documents; however, some re-clustering should occur after 25%-50% growth. Chapter VII develops a model of a disk storage algorithm and suggests storing the hierarchy by levels for most efficient access. Chapter VIII describes a scheme for query alteration during searches which uses term-term relationships in profiles. Chapter IX indicates that a clustered file uses no more space than an inverted file and provides more flexible search criteria. Chapter X is a summary of findings.

In total, this thesis attempts to answer the question "Is a clustered file organization suitable for on-line document retrieval?". The proposed organization compares favorably in terms of speed and storage economy; various request-document matching procedures, search strategies, and feedback schemes are easily implemented. Search precision is less, but compensated by a flexible level of recall (low or high). Furthermore, arbitrary accesses for individual records are not required since those records with a high probability of satisfying a request are concentrated in a few disk locations. Therein lies the greatest value of a clustered file.

Synopsis

This dissertation examines the file organization problem in an on-line, computerized document retrieval system. Its aim is to demonstrate the utility of a clustered file in such an environment. The clustered scheme uses a classification algorithm to partition a document collection into overlapping sets of related items. The documents in each cluster are stored contiguously and accessed through a hierarchy of profile vectors which "summarize" cluster content. There are several reasons why this approach is superior to conventional file organizations based on chains or inverted indices. First, all data for each document are stored only once and in the same location. This allows the use of any query-document match function and the implementation of advanced features such as relevance feedback searching and dynamic document space modification. Second, since there are no pointers or lists, the storage overhead promises to be low. Third, and finally, searches are economical and flexible since most clusters are eliminated during query-profile matching. Once a cluster is chosen for detailed examination, its contents are accessed rapidly since they reside in contiguous locations. Consequently, it is not the case that retrieval times and costs increase linearly with the number of documents examined nor with the query length.

The experiments in this thesis are concerned with the profile hierarchy--its construction, use, and maintenance. These topics are

important since without accurate, economical profiles, cluster generation is much more costly and the file cannot be searched properly.

Chapter I contains an introduction to document retrieval, file organization, and the problems to be investigated. It is argued that an on-line retrieval system is required in order to overcome incomplete and inaccurate text analysis and to allow users to probe data bases, refine their requests, and control searches. Concerning implementation, the choice of file organization is the primary factor in determining how information is accessed and how the costs are distributed. A clustered file is suggested as being flexible and powerful enough to handle the diverse document search criteria and requirements for information displays, while having favorable cost-performance characteristics.

Chapter II surveys current file organizations, particularly the manner in which they partition the file and provide linkage among related records. Five general schemes are compared--sequential, chained, inverted, computer-access, and clustered. The results indicate that only the inverted and clustered organizations provide the speed necessary for on-line document searches. Both allow implementation of relevance feedback and document space modification; however these options require additional storage space in inverted files. Generally, clustered files support more flexible search strategies, but retrieve with somewhat less precision. These tradeoffs are explored fully in later sections.

Chapter III contains a thorough discussion of clustered files and background information for the experiments. Classification methods, search strategies, and query clustering are considered briefly. The main topics are the construction, use, and maintenance of profiles. Three types of standard profiles are presented: P_1, having no term weights; P_2 having weights based on document frequencies; and P_3 having weights based on total term frequencies. These are compared to Doyle's rank value profiles, P_r. When additions are made to the file, maintenance is viewed in terms of small adjustments to keep profiles in the "center" of their clusters. However, ultimately, the collection must be re-classified; the frequency of this operation is to be determined. Other important matters include limitations on profile length and the order of storage (level, subtree, hier-filial). Both of these help determine search times and storage overhead.

The experimental environment is described in Chapter IV, including the characteristics of the SMART system, the Cranfield document collection (1400 abstracts, 225 requests), and the three clustered files. Two evaluation schemes are presented. One is the regular SMART procedure based on precision and recall data for fixed search strategies. The second is a new, economical method which is independent of search strategy and accounts for system effort more accurately. It is based on measurements of recall ceiling (percent of relevant that are recoverable) and precision floor (percent of total recoverable that are relevant) taken before each cluster is expanded. Naturally clusters are expanded in order of decreasing similarity with the request vector. Both methods are used in Chapters V to IX.

Chapter V reports on an extensive set of experiments with profiles, in particular, standard and rank value vectors, biased search results, vector length, and frequency and weighting considerations. The best profiles are those with term weights based on frequency ranks and hence are a compromise of the standard (P_2 or P_3) and rank value types. The use of ranks keeps the range of weights small and reduces correlation domination. Keeping the weight origin at a minimum eliminates bias and maintains maximum distinction among terms. The success with ranks indicates that the importance of terms does not increase linearly with frequency, but in a more gradual way (approximately logarithmically). This accounts for the success observed with profiles using categories of weights and the partial success of unweighted profiles with deletion of "noise" terms. The tests also indicate that a large portion (80%) of low weighted terms can be eliminated with only small degradations in search performance. High frequency significant terms cannot be deleted, moved upward in the profile hierarchy, or assigned smaller weights than less frequent terms. Overall, the results indicate methods for constructing reasonable and economical profiles, but indicate that further improvements can be made.

The file maintenance experiments in Chapter VI are summarized in two findings. First, if reasonable profiles are used such as those described in Chapter V, then a clustered file may increase its size

25%-50% before the degradation in search performance is such that reclassification is required. These percentages are calculated as the ratio of additions to the current file size. Second, adjustments to profiles during updating are of only slight benefit. The most reasonable scheme is to adjust the weights of only existing terms and not to introduce new terms (ALTER option). Since the adjusted profiles keep the same size, they can overwrite their predecessors without destroying any storage organization in the profile hierarchy.

Chapter VII describes experiments involving the use of indexed sequential access for managing a disk resident clustered file. For forward search strategies, storing the hierarchy by levels is found to provide the most rapid response. Furthermore, it is shown that cluster searching can retrieve many of the relevant documents obtained in a full search, but at much less cost. For example, cluster searches requiring 10-16 disk accesses achieve about 70% of the precision and recall values of a full search requiring 65 accesses.

The basic idea of the query alteration procedure in Chapter VIII is to associate a small thesaurus with each profile and to expand selected request terms during searches. Each mini-thesaurus reflects the pecularities of the vocabulary in its cluster. Consequently, there is a unique opportunity to combine the use of broad, general term relationships on upper hierarchy levels and specific, local relationships on lower levels. Unfortunately, for the options tested, query alteration is of doubtful value when employed automatically.

However, it might be used profitably as part of a negative feedback procedure.

Chapter IX compares the storage requirements, speed, and effectiveness of clustered and inverted files. The inverted organization requires twice as much storage space as a clustered file in order to provide equivalent retrieval services. For a specific number of disk accesses, the inverted search retrieves a fixed number of documents and generally achieves high precision at a specific recall level. With the same effort, a cluster search provides many or few documents. Although its precision is less, the search may have a recall level which is higher or lower depending on the number of retrieved documents. This flexibility within searches of a single cost figure is considered a genuine advantage.

The final chapter summarizes the work. The clustered file organization is considered more advantageous than other file organizations since it allows greater flexibility in matching and searching without increased retrieval time or storage costs.

Chapter I

Introduction

1. Automated Information Systems

A. General System Types

Information management is of increasing concern in the modern world. Commercial, scientific, educational, and governmental institutions produce and distribute such large quantities of reports, research data, and other literature that it is difficult to keep abreast of almost any field using manual processes. Automated information systems--combining high-speed computing equipment, mass storage devices, and sophisticated programming systems--appear to be one way of containing this information explosion.

Hayes (1) distinguishes three types of systems based on their scope of activities--data base, reference, and text processing. Data base systems manipulate files of fixed-format records and generally provide capabilities for adding-deleting records, changing the contents of selected fields, and retrieving items with specified properties. A familiar example is that of an airline reservation system in which passenger records contain the name, flight number, destination, time of departure, and so forth. In the course of a day's activity, many records are created, deleted, and changed in order to reflect current business conditions. Queries to the file are expressed as logical combinations of special keywords and commands which the system is designed to interpret. In this respect, the operations are oriented toward experienced personnel rather than a general public. Data base systems have wide use currently, and a number of generalized software packages are available to meet most commercial needs.

Reference systems deal with more complex data structures such as printed text or pictures, and retrieve references to items rather than actual articles or photographs. The information content of a text or picture is contained in a set of manually assigned descriptors or keywords; sometimes automatic dictionary procedures are used as indexing aids. The same record processing facilities are available as in data base systems, although retrieval conditions may be relaxed from the strict criteria of a Boolean search formulation. In particular, a simple function of the number of matching request descriptors could determine which file items are closely related to a request. Subsequently only the highest scoring records are actually obtained for user inspection. Since highly structured queries are not needed, requestors should find reference systems easier to use than data base systems. The NASA, Medlars, and Chemical Abstracts retrieval services are examples of reference systems dealing with aeronautical, medical, and chemical literature.

Full text processing systems include automatic statistical, syntactic, and semantic procedures to format intricate data structures containing the implicit and explicit information in the original text. The search process is complex also, including correlation measures and linguistic processing, as well as man-machine interaction, data displays, and iterative searching. If complete text is stored, fact retrieval may be possible so that answers to questions are given rather than lists of references. At this time, information systems which approach full text processing--SMART, SIR, and STUDENT--are still in their experimental stages.

B. Document Retrieval Systems

A document retrieval system is a combination of reference and text processing systems usually limited to simplified content analysis (dictionary or thesaurus), but containing complex search negotiation procedures. Syntactic and linguistic methods are often avoided for cost-effectiveness reasons. As in reference systems, document identification numbers are retrieved although citations, abstracts, or even full text could be printed if storage provisions allow. In the particular model used for this study, the subject content of a document is reflected in a set of weighted keywords derived automatically from the original text. Natural language queries are indexed in the same fashion and matched with file items using a correlation function. Those documents with the highest correlations have their accession numbers returned to the user. If more precise or complete information is desired, the search is continued using a feedback or other strategy. Additional information about the experimental system is given in Section 3 of this chapter and Section 1 of Chapter IV.

The application of computer systems to document retrieval raises many intellectual and technical problems. Automated processing obviously substitutes software logic for the human intellect available in a good reference library. Regardless of their complexity, all programs operate without a genuine understanding of text and therefore are inaccurate by human standards. Undoubtedly, the overall goal of information retrieval is the development of methods which are equal to human ingenuity with respect to retrieving relevant documents; however, research aimed at obtaining good procedures is complicated not only by the difficulty of text analysis, but also by the nature of relevance itself. Since each user is the judge of whether a document is relevant to his request, it is difficult to find measureable properties that always discriminate among documents properly. Moreover, users often do not have well defined information needs, but still require the system to retrieve specific data in answer to vague questions. Lastly, even if a query statement is precise, it may not correspond to the stored text in any reasonable way. To overcome these problems, a great many automated systems operate on an interactive basis. The system supplies tutorial instruction, information displays, file statistics, and iterative processing while users clarify ambiguous terms, expand or refine queries, and identify relevant or non-relevant documents in feedback processes. This combination of computer hardware for rapid processing and user supplied semantic inputs forms an integral part of the system design considered here.

The most important implementation matter associated with automatic document retrieval systems is the selection of a file organization. In nearly all instances the size of the data collection requires the file to be structured so that only part of it is manipulated for any operation. While the previous problems deal with semantic and human factors, file organization is the primary technical issue, involving interactions among file characteristics (size, complexity), user satisfaction (response time), software complexity (search, interaction, maintenance), and hardware (processors, storage media). This thesis examines the file problem in an on-line document retrieval system with respect to querying the file by subject content rather than bibliographic keys such as author, journal, etc. Its aim is to demonstrate the feasibility and utility of a particular organization--a clustered file--in this environment.

2. File Organization

A. General Purpose and Definition

The purpose of a file organization is to provide convenient and efficient use of stored data. Convenience relates to the ease of retrieving records, implementing desired software, and maintaining the file. Other factors include the ability to extend to future applications and to take advantage of natural structures within the data, patterns of usage, and special features of operating systems. Measures of efficiency generally pertain to memory space (amount of waste, overhead, and redundant storage) and access time (overhead computations and I/O operations). User satisfaction may or may not be an evaluation criterion because of its subjective nature and close relation to the convenience of providing software services. Document retrieval systems, however, often measure user satisfaction by the quality of the retrieved material. Lefkovitz (2) draws the following distinction between file structure (organization) and information structure. File structure denotes the record layout, directory setups, and file partitions necessary to meet specifications for access times, storage economy, and maintenance effort. Information structure is an inherent property of the data that exists by design or by the way the data appear in the collection. This structure, whether natural or imposed, can be used as a basis for partitioning file items into groups. For example, items with a hierarchical information structure exhibit superior-inferior relationships and can be grouped accordingly; records in an associative structure might be grouped if they share common properties. The file structure may or may not take advantage of the information structure.

Another way of viewing file organization examines only its major purpose--data access. With respect to retrieval programs--search, display, maintenance--the information structure divides the file into groups of items which are logically related and processed together. The method of logical access determines how these related items are associated with each other. For example, associations might be implemented by chains, lists, naming conventions, physical adjacency, or functional relationships among record identifiers. With respect to the operating system, a file organization includes a method of physical access to facilitate locating records on actual storage devices. For example, if absolute addresses are used, records are located by direct access. However, if names, reference numbers, or relative positions are used, then sequential, indexed sequential, or partitioned access may be applicable (3). Consequently a file organization specifies two interfaces:

1) between retrieval programs and the supervisor's I/O system (logical access) and
2) between the I/O system and storage devices (physical access).

The first interface is equivalent to establishing a file directory for use in search negotiation. It is of primary interest in this research because of the rich supply of data structures applicable to such directories and because of the challenging nature of access by subject content.

The second interface is given less attention because its options are generally limited to those supported by a manufacturer's software.

B. File Organization in a Document Retrieval System

In all, a document retrieval system manipulates a variety of data including dictionaries, thesauri, search vectors, citations, and abstracts. Each of these may exist as a separate file or several may be combined in an integrated file with a single access method and multiple directories. In particular, dictionaries and thesauri are often combined for use by content analysis routines and search vectors and retrieval data may be integrated for search negotiation. In either case, processing generally proceeds through the files one at a time starting with a dictionary lookup of query terms and ending with the display of retrieved document citations. However because of interactive processing, a file may be accessed several times in succession to display information, process an altered query, or enter new data. Part or all of this cycle may be repeated during iterated searches. In an on-line environment, all processing must occur fast enough to satisfy user impatience; only file maintenance is considered an off-line operation because of its non-critical nature.

For this study, a document file is defined as containing:

1) the searchable descriptions of texts (document vectors),
2) initial retrieval data (reference numbers or short citations), and
3) whatever directories are necessary.

Primary attention is given to the problem of logical file organization with respect to subject searches, that is, logical access. Queries involving only bibliographic information are assumed to be processed through separate directories. Even limited to subject access, the organization task is complicated by adverse file characteristics (enormous size, rapid growth, and variable length records), the demands of sophisticated programs operating on-line, and the elusive nature of relevancy. It may be recalled that relevancy is a user defined property and is not necessarily limited to records having the same descriptors as the query. As a result, logical access includes defining connections among items with loose semantic relations (imposing an information structure) as well as providing a directory for accessing them.

The previous section sets forth the purpose of file organization in terms of convenience and efficiency. With regard to on-line document retrieval, a few design criteria must be emphasized or added. First, the absolute prerequisite is real-time response. Although, it may be possible to process several requests simultaneously (batching), the system must be prepared to treat users independently since it is unlikely that any two requests in a batch pertain to the same portion of the file. Second, user-machine interaction is definitely necessary to clarify and satisfy information needs. This includes presentation of data and processing sensitive to user responses; examples are tutorial instruction, query formulation aids, browsing, query alteration processes, and feedback (iterated) searching. Third, query-document matching is to be based on a correlation function rather than satisfying a logical predicate or simple coordinate matching. Either of the latter methods could be used; however, correlation functions provide greater flexibility in their treatment of both queries and documents. Specifically, a change in the function or its parameters may allow more strict or lenient scoring, different emphasis on vector properties, or the addition of new factors to the entire procedure. Fourth, an appropriate cost-performance tradeoff is desirable to satisfy users wanting small amounts of information at low cost and those willing to pay for comprehensive searches. Finally, evaluation standards must include not only time and space, but user satisfaction. The standard measures of _precision_ and _recall_* are used to evaluate the quality of retrieved material and thereby judge user satisfaction.

A complete set of design characteristics and evaluation measures for file organizations undoubtedly includes additional aspects of information systems. However, those outlined so far are the most important and serve as a basis for comparisons in this work.

3. The Document Retrieval Environment

In order to be explicit about the type of retrieval system envisaged, the following sections contain descriptions of the document file, user factors, software services, and disk storage devices.

A. Document File Characteristics

A document retrieval system operates on a collection of natural language articles indexed in some fashion and stored as searchable vectors. The document file itself is characterized by its large size, rapid growth, and variable length records.

---

* Considering the documents retrieved by a search, _precision_ is the percent of retrieved which are relevant and _recall_ is the percent of relevant which are retrieved.

With respect to file size, it is noted that storage for the Library of Congress catalogue would have required 10^{12} bits in 1962. Further, a number of public and university libraries already contain in excess of a million volumes and the general trend is for holdings to double every 15 years (4). Even a modest system for handling journal articles in a specialized field might expect several thousand additions per year (5). These facts clearly indicate the need for mass storage devices and, for really large files, several classes of storage media--disk, data cell, photostore, tape.

In addition, decreases in file size are rare because future editions of texts or re-publications of articles are generally viewed as new items rather than as replacements for old ones. In situations where current literature is of primary importance, the document file may be segmented into active and archive storage. However, this does not provide genuine reduction in many cases since all data remains retrievable and directory entries for older items are transferred in and out of memory during general processing. In most information centers the files only increase in size as new items are entered during periodic off-line maintenance runs. Usually, there is no significant demand for a real time update capability.

Texts obviously require variable length records because of their own variable lengths, the automatic indexing process, and the amount of retrieval data. Usually a variety of standard access points are included in the vector--author, publication date, journal or call number, headings--

as well as the set of index terms for describing subject content. In all, 200-300 characters are not unusual lengths for document records.

B. User Factors

Several user situations must be anticipated in the design of a successful on-line retrieval system. First, at least two classes of users must be treated; new or infrequent users demand simple, straightforward query submission, search, and retrieval functions while experienced persons need flexible and powerful procedures. Second, the system must be able to cope with vague questions since in many cases a user in either class is unable to state his information needs accurately. Third, even if a precise query statement is given, it may not correspond to the stored text in any reasonable way. This is especially true if important query terms have extremely high or low frequencies. In the former case, the search may yield too many items and in the latter case not enough items. Fourth, only some of the retrieved documents will be relevant due to inaccuracies in the indexing process, matching functions, etc. This condition necessitates some form of query alteration and iterated search if more relevant information is to be obtained.

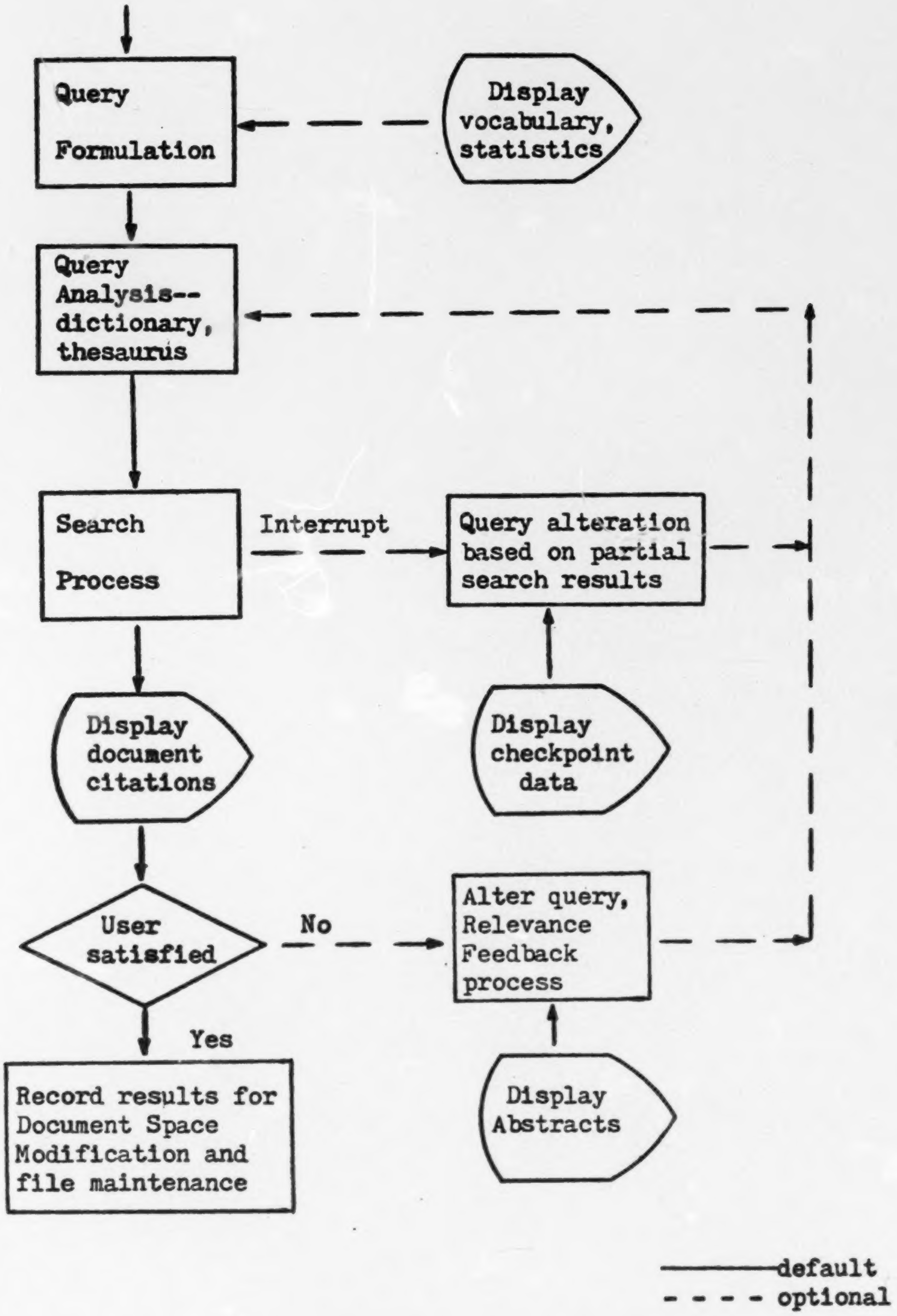
In an on-line environment, many of these problems are solved through user responses to information displays. For example, during query formulation, synonyms for the original query words might be presented along with frequency information in order to facilitate selection of proper terms. Or the search process might be interrupted before its completion and early results examined to check that appropriate search paths are followed. Once the search is completed, citations or text for the retrieved documents can be viewed. If the results prove unsatisfactory, new searches could begin immediately. (See Section I.3.C)

To summarize, the success of a document retrieval system requires on-line operation to overcome various user factors as well as incomplete text analysis. For these reasons, this research investigates file organizations which are suitable for on-line systems with respect to cost and flexibility in the retrieval process.

C. Software Services

The most satisfactory way to accommodate users is through on-line operation and options for selecting a wide variety of processing methods. A typical search negotiation in such a system is described by the flowchart in Figure I-1. The overall requirements for query formulation aids, document display or browsing, and simple query alteration have been mentioned. The search, relevance feedback, and space modification functions require additional explanation.

Subject searching involves several phases of activity. To start, assume that a query has undergone dictionary, thesaurus, or other analysis and that its content is represented by a vector of index terms and associated weights. The next step is to calculate correlations between the query and the most promising documents. How this is accomplished depends on the file organization. Recalling that most file systems partition records into logically related groups, the usual procedure is to examine all the items in several groups. For example, consider a file using chains to link documents having the same keyword. The chained records form partitions whose elements are located by following sequences


Query Formulation
Display vocabulary, statistics
Query Analysis--dictionary, thesaurus
Search Process
Interrupt
Query alteration based on partial search results
Display document citations
Display checkpoint data
User satisfied
No
Alter query, Relevance Feedback process
Yes
Record results for Document Space Modification and file maintenance
Display Abstracts
default
optional

Typical Search Negotiation in an

On-Line Document Retrieval System

Figure I-1

of pointers. To search the file, some or all of the chains named by query terms are followed and a correlation is computed for each item encountered. Similar situations occur for serial, inverted, calculated, or clustered organizations. (See Chapter II.) In any case, once all correlations are gathered and sorted by value; the names, citations, or abstracts for the highest scoring documents are obtained. At this time, a set of secondary restrictions may be applied before a presentation is made to the user. Restrictions often include bibliographic or quantity limitations on output, verification of word order within the text, and others. Regardless of the secondary processing, the major portion of the search effort involves accessing file partitions and accumulating correlations. These are the significant aspects as far as file organization is concerned.

Relevance feedback is a tool for obtaining improved search results through iterative searching (6, 7, 8). Briefly the process works as follows. A user at an on-line console views the citations or abstracts of a few documents retrieved by an initial search. Immediately or after consulting the source texts, he enters decisions as to which items are definitely relevant or non-relevant, possibly leaving a few items unjudged. The system alters his original request by adding or emphasizing descriptors found primarily in the relevant documents and by removing or reducing the importance of terms found primarily in the non-relevant. A new file search is made, the results presented to the user for his inspection, and the entire process repeated if desired. Several variations of this general scheme are possible. Positive feedback employs only terms from the relevant documents whereas negative feedback uses only the non-relevant. These may be used separately, jointly, or selectively depending on the output of the initial search. Positive techniques are especially valuable when some relevant are retrieved and additional similar items are desired. Negative methods produce some success even if no relevant are retrieved in the initial search or if the relevant obtained are dissimilar. If a clustered file is used, documents from separate clusters may generate distinct queries for operating within each cluster. Such query splitting procedures have been investigated by Borodin, Kerr, and Lewis (9). Although evaluation is difficult, at least one investigation of feedback searches reports improvements of 5% to 10% in precision and recall, the standard performance measures for information systems (7, 8). Thus feedback has been shown to be an effective retrieval tool in an on-line environment and should definitely be included among the software services.

Document space modification tries to pass on the success of previous searches to future users by making the retrieved relevant correspond more closely to the original query (10, 11). Considering the set of documents obtained after feedback and all other processing, each relevant item is modified according to whether its terms appear in the document, the query, or both. A similar, but inverse modification is applied to the retrieved non-relevant so that there are positive and negative strategies here also. Experiments by Brauen indicate substantial improvement in future performance based on these methods and indicate their importance to current retrieval systems.

Both techniques affect file organization in that they require access to entire document vectors. Some file schemes distribute the index terms of a vector throughout memory so that its entirety is unretrievable in a single access. This may be acceptable for the search process, but in order to use feedback or space modification, complete vectors must be easily obtained.

D. Storage Devices

The discussion of document file size points out that considerable memory space is required for even a moderate size retrieval system. In general, direct access devices have the most appeal in the solution of the mass storage problem. Disks and drums provide sufficient capacity and speed to warrant their use for dictionaries, directories, and document vectors. Text and occasional items may be assigned to data cells, photostores, or magnetic tape.

Throughout this research, the IBM 2314 Direct Access Storage Facility is used as a model device. Its hardware consists of eight interchangeable disk packs (volumes) each with a capacity of 29 million characters (12). The average access time is approximately 100 milliseconds for a 3600 character data block providing there is no queue for the single I/O channel servicing all packs. This also excludes whatever delays are encountered while unblocking records. Specifically the access time consists of waits for the completion of four events:

1) access motion--positioning the access arm (set of read-write heads) at the proper cylinder,
2) head selection--switching to select the read head for the appropriate track,

3) rotational delay--rotation necessary to position the read head at the start of the data, and
4) data transfer--reading the data into main memory.

As a result of the mechanical motions involved, disk fetches require several milliseconds and are slow and expensive relative to internal computing speeds. Cylinder changes are most costly as a substantial amount of time is required to accelerate the access arm. For this reason it is better to input a single large data record than several small ones scattered throughout the volume. Other delays are much smaller, the head selection time being negligible. A full revolution of the pack requires 25 ms regardless if data is read. Consequently, the expected rotational delay is 12.5 ms and the transfer time is proportional to the amount of information moved. Table I-1 summarizes the storage and access specifications for the IBM 2314 (12).

There are many ways of reducing the delays from disk fetches. Accesses can be confined to the same cylinder if possible or programs might be implemented so that processing on the current record is finished in time to fetch the next record while in the same rotation. However, the best way to insure short, real-time response is to minimize the number of fetches. This holds true even in time-shared or multiprogrammed systems, for a small number of I/O interrupts means that program execution is suspended less often. As a result, the primary measure of response time and the quantity for minimization is the number of disk accesses.

Storage Capacity	
Packs/Device	8
Cylinders/Pack	200
Tracks/Cylinder	20
Characters/Track	7294
Characters/Cylinder	146K
Characters/Pack	29.2M
Retrieval Times	
Access time	
Minimum	25ms
Maximum	135ms
Average	75ms
Rotational delay	
Maximum	25ms
Average	12.5ms
Transfer rate (per character)	.0032ms

Storage Capacity and Retrieval Time

for the IBM 2314 Direct Access Storage Facility

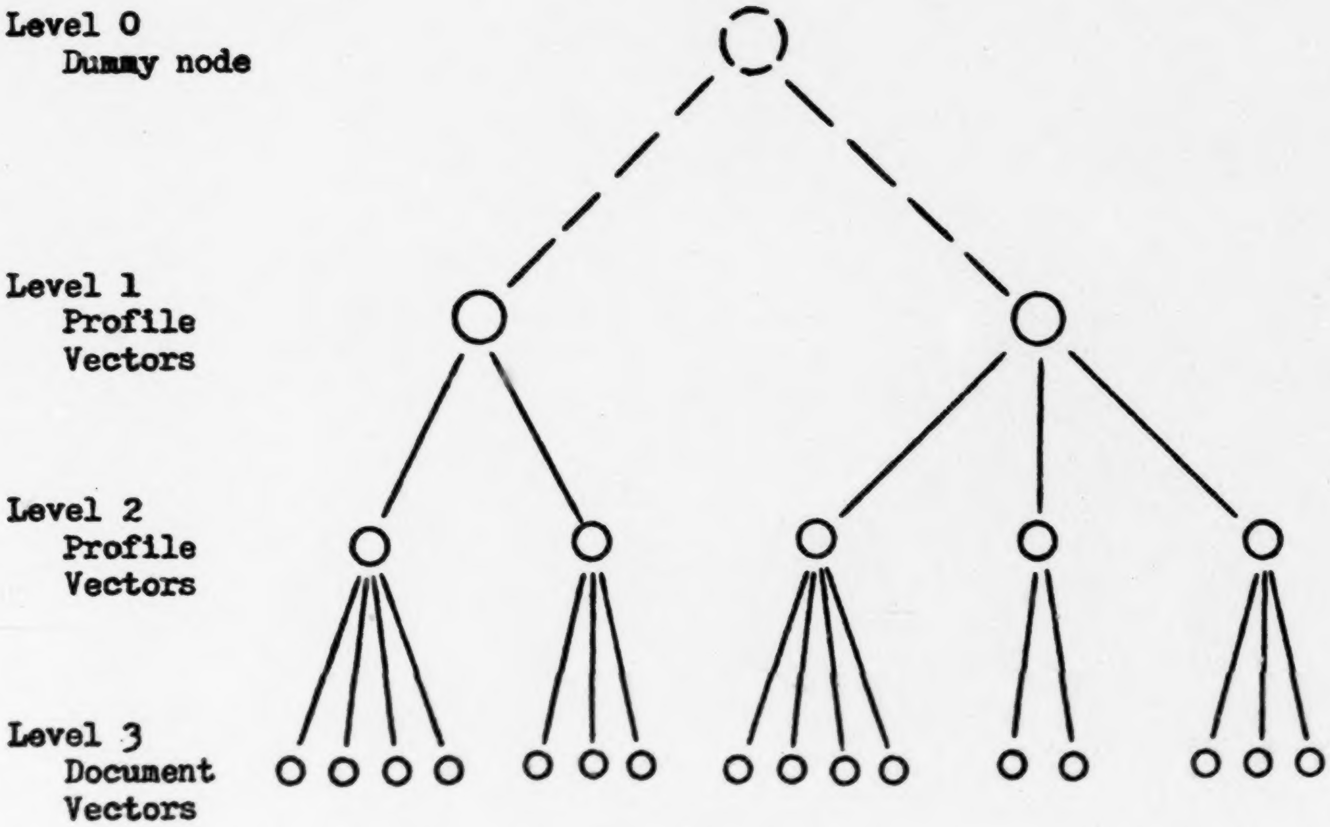
Table I-1

4. The Clustered File Organization

A. General Concepts

A clustered file is arranged so that similar documents are located near each other within the storage medium. Generally an automatic classification procedure is used to compare document descriptions with each other and define the so-called clusters. Retrieval programs either process all or none of the items in a cluster and therefore transfer large blocks of data from mass storage rather than many smaller individual records. For each cluster, a profile characterizes its information content and acts as a directory element. Loosely speaking, a profile is an aggregate of the index terms in the clustered documents and has a structure resembling a document or query vector. The search process matches a query with each profile and examines in detail only those clusters with the highest scores. For large collections, there may be a great many clusters (profiles) and the classification procedure is often re-applied to group profiles. The result is a hierarchical directory similar to that in Figure I-2.

Referring to Lefkovitz's ideas (2), the classification imposes a hierarchical information structure on the data and the tree storage and search procedures are parts of the file organization. A hierarchical structure is natural to a document file with respect to usage since users are able to find analogies between the automated search and their personal library activities. In an interactive environment, a user might browse within the file by viewing information in the profiles. Subsequent operations could allow query alteration before or even during the search. Finally if at least one relevant document is located, additional



Sample Hierarchy Resulting from Document Clustering

Figure I-2

pertinent articles should be found in the same file area.

B. Areas of Investigation

This thesis treats three aspects of a clustered file organization--profile definition and storage, updating, and secondary uses of the hierarchy. Because profiles are aggregates of document vectors, their definition involves many factors including limitations on the number of index terms, weighting procedures, and number of allowable sons. All of these affect the system's ability to discriminate among clusters and hence the accuracy of performance. The actual order of tree storage strongly influences response times and search cost. File update techniques are important in order to maintain speed and accuracy. New items must be re-distributed occasionally to achieve physical proximity of related information and thereby maintain reasonable data access times. In addition, profiles might be modified during update in order to reflect the presence of new cluster members. A few additions make little difference, but it is unreasonable to expect a hierarchy to perform well after the file has increased its size several times. Finally, the high cost of current classification methods leads to the idea of using the profiles and document hierarchy for several purposes. By spreading the clustering expense among all applications, this overhead is more easily justified. Several possibilities are discussed, including the concept of associating a small term theasurus with each node in the hierarchy. Thus, statistically related thesaurus terms are available for modifying requests as the search enters various parts of the collection.

These are the problems for consideration and all of them are related by their association with the cluster profiles--either definition, storage, changes, or secondary uses. Actual classification methods are not examined here, but surveyed in Section III.2. Search techniques are described only to the extent that they are used in the research.

5. Outline

The present chapter outlines the purpose and operation of a document retrieval system, the file organization problems within it, and the general processing environment in which these problems are to be solved. The next two chapters concentrate on file organization methods, first surveying current schemes, then describing the clustered file in considerable detail. Chapter IV is devoted to the experimental system and evaluation methods used in this research. Chapters V to VIII are given to the areas for investigation--profile definition, updating, tree storage, and query alteration based on profile information. The final chapters include a comparison of the clustered and inverted organizations and the conclusions and future investigations suggested by this work.

References

1. R. M. Hayes, Information Retrieval: An Introduction, Datamation, March, 1968.

2. D. Lefkovitz, File Structures for On-Line Systems, Spartan Books, New York, 1969.

3. IBM System/360 Operating System, Supervisor and Data Management Services, IBM Corporation, White Plains, New York, 1968.

4. I. A. Warheit, File Organization for Library Records, Journal of Library Automation, Volume 2, No. 1, March, 1969.

5. M. Davis, D. Murray, File Organization for an On-Line Retrieval System, Unpublished class report, Computer Science 635, Cornell University, January 1969.

6. J. J. Rocchio, Document Retrieval Systems--Optimization and Evaluation, Harvard University Doctoral Thesis, Report ISR-10 to the National Science Foundation, Chapter 3, Harvard Computation Laboratory, March 1966.

7. E. Ide, Relevance Feedback in an Automatic Document Retrieval System, Report ISR-15 to the National Science Foundation, Department of Computer Science, Cornell University, January 1969.

8. G. Salton, Interactive Information Retrieval, Technical Report 69-40, Department of Computer Science, Cornell University, August 1969.

9. A. Borodin, L. Kerr, F. Lewis, Query Splitting in Relevance Feedback Methods, Report ISR-14 to the National Science Foundation, Department of Computer Science, Cornell University, October 1968.

10. T. Brauen, R. Holt, T. Wilcox, Document Indexing Based on Relevance Feedback, Report ISR-14 to the National Science Foundation, Department of Computer Science, Cornell University, October 1968.

11. T. Brauen, Document Vector Modification in On-Line Information Retrieval Systems, Report ISR-17 to the National Science Foundation, Department of Computer Science, Cornell University. September 1969.

12. IBM System/360 Component Descriptions, 2314 Direct Access Storage Facility, 2844 Auxiliary Storage Control, IBM Corporation, White Plains, New York, September 1969.

Chapter II

Survey of File Organizations

1. Introduction

Most file organizations partition their records into groups related by some natural or imposed criteria. The logical access method specifies how related items are associated while the physical access method facilitates translating record names into storage addresses. Here, file structure and logical access are emphasized rather than the details of data management services (physical access). This chapter surveys existing file organizations applicable to a document collection and evaluates their utility in an on-line retrieval system. In these systems, a search consists of at least four steps.

1) directory scan--choosing the file partitions for detailed examination;
2) accumulation of query-document correlations within the selected partitions;
3) selection of items for output--sorting correlations, fetching retrieval data, and applying secondary restrictions; and
4) information display.

Particular attention is given to the directory scan and the generation of correlations since these are the most crucial and costly steps.

In all, five types of organizations are discussed: sequential, chained, inverted, calculated access, and clustered. Some variations and combinations are considered also. Much of the terminology is adapted from previous surveys by Lefkovitz (1), Salton (2), Lowe and Roberts (3), and Meadow (4). As far as evaluation is concerned, most surveys examine retrieval time and storage usage. In addition to these, the present discussion considers the evaluation quantities outlined in Section I.2.B--convenience of implementing desired software features, maintenance, quality of retrieved material, cost-performance tradeoff, and general appropriateness to on-line document retrieval.

2. Methods of Logical Organization

A. Sequential Files

A sequential file organization stores documents in the order of their acquisition and retrieves them by a complete scan of all records. In fact, the file shows little organization at all since it has no partitions, no directories, and no particular order for item storage. Because the complete file is scanned during a search, the retrieval time is prohibitive for on-line operation. However, sequential files are still justifiable in current awareness systems or in other situations where it is possible to accumulate requests over short periods of time. In these cases, a search is made when the query batch is large enough to yield an acceptable cost per user. A number of retrieval systems use the sequential organization and employ this batching technique (5, 6, 7).

In spite of their large access time, sequential files have several favorable properties. First, their storage requirements are minimal, since no pointers, linkages, or other overhead is involved. Second, information can be retrieved via almost any criteria since entire document vectors may be examined. For example, bibliographic information;

the presence, absence, or weights of index terms; and almost any correlation function can be used for query-document matching. Third, maintenance is trivial since new records are simply stored at the end of the current file.

Finally, the expense of structuring the file in any other way may force the use of the sequential scheme. This is obviously the case if the volume of activity is so low that the cost of organizing the file exceeds the total cost of the searches made. Coffman and Bruno recognize this situation and suggest splitting the file into structured and sequential parts (8). The search process begins in the structured portion and proceeds to the sequential portion only if necessary. Items retrieved from the sequential subfile are subsequently transferred to the structured section. A slightly more complicated scheme would record the number of times a document is retrieved and retire information from the structured subfile when its retrieval rate falls below a chosen threshold. Exactly where the file should be split is unclear. Lipetz considers the tradeoff point between sequential and some other organizations; his results may have bearing on this problem (9).

A similar scheme proposed by Leimkuhler partitions the file into bins so that the probability of a successful search is maximized for the effort expended (10). Each bin forms a sequential subfile of the next group of documents judged equally likely to satisfy queries. The search is cumulative; always starting with the same initial bin and proceeding to the last bin, unless the user is satisfied earlier. The points of file division and expected search effort are computed from a form of Bradford's law relating literature productivity, number of references per document, and collection completeness. Leimkuhler favors a 2-bin system having 20% of the most pertinent documents in the first bin. It is expected that 2/3 of the requests are satisfied by searching only the initial subfile.

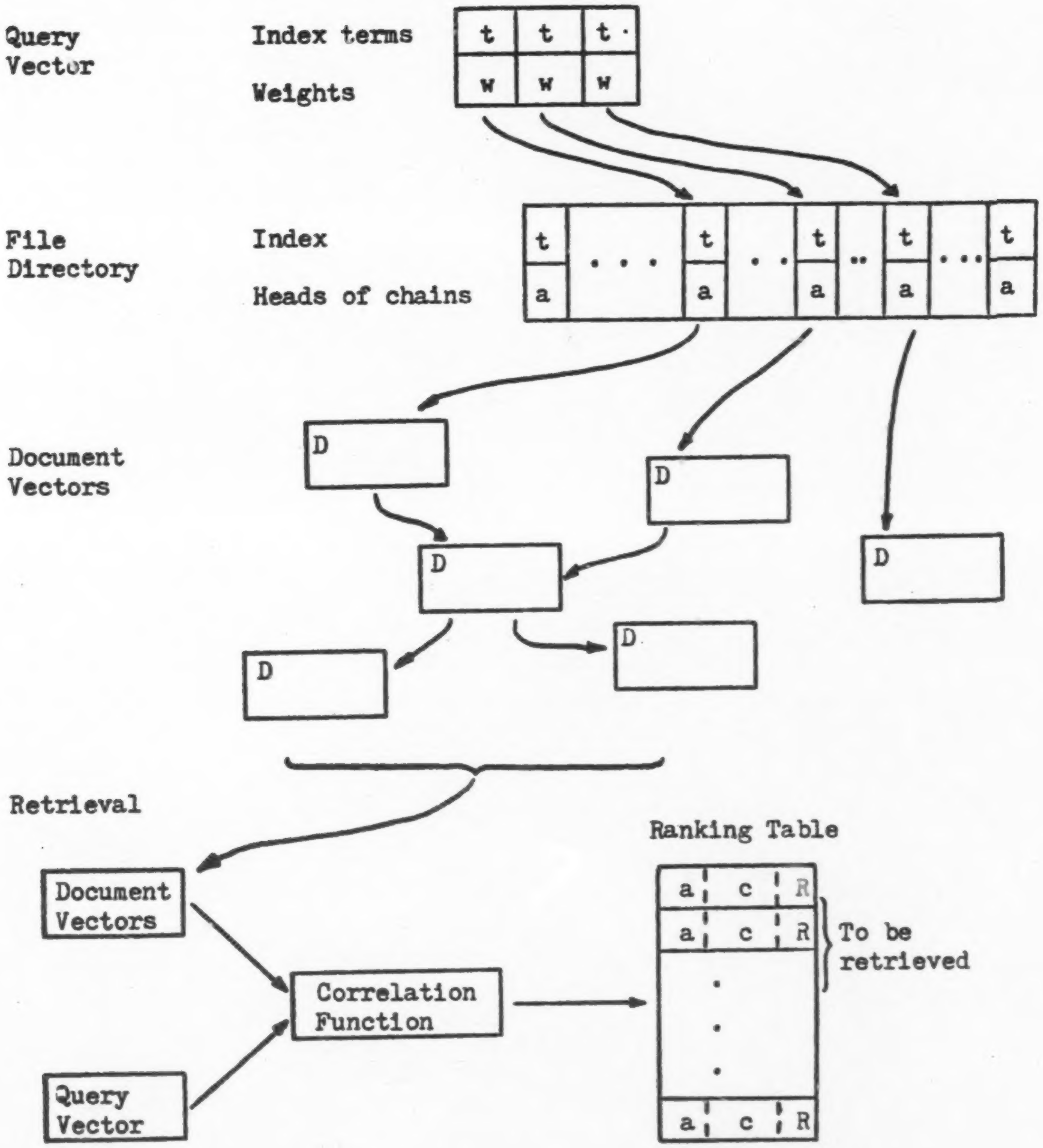
In a more restricted situation, Ghosh (11) considers searching a sequential file to find documents containing all the descriptors found in the query (not necessarily implying true relevance). A set of queries is defined to have the consecutive retrieval property (CRP), if there is a sequence for storing all records so that the documents satisfying each query are located consecutively. A file ordered in this way has not only minimum storage requirements, but also minimum search time if a suitable directory is used. Ghosh shows that an arbitrary query set can always be divided into overlapping groups of queries with CRP. The complete file, then, is the concatenation of sequential subfiles associated with each query group. Although this organization appears interesting and perhaps applicable to finding actual relevant documents, no implementation or evaluation information is known.

These extensions of sequential files do not change the fact that their access times are too slow to permit on-line search, display, feedback, etc. The results indicate, however, that sequential organizations cannot be discarded too quickly, especially when used in conjunction with other techniques. The schemes of Ghosh and Leimkuhler actually border on single-level document clustering. The fact is, that serial order is acceptable for parts of a document collection, but not its entirety. The problem is to decide how the file should be partitioned and how each portion should be accessed.

B. Chained Files

Logically, the chained file organization partitions documents into sets of vectors having common index terms. The elements of each set are chained together and accessed by following a sequence of pointers. The file directory consists of a table giving the head of each chain while actual document vectors are stored in order of acquisition. A new document is attached to the file by placing it at the head of the chains corresponding to its index terms. As a result all chains are in decreasing order by accession number and point to the most recent information first. The file search procedure using this type of organization is depicted in Figure II-1. A binary search, hashing method, or other lookup technique is used to scan the directory and locate the beginning of the document chains corresponding to each query term. Subsequently all items along these chains are fetched and correlated with the query vector to establish which documents are suitable for output. Once the retrieval cutoff is chosen, accession numbers or other retrieval data are displayed. In some situations a separate file is accessed in order to present complete citations or abstracts for the documents at the top of the ranked list.

With a chained organization, a document vector is a set of triples (t,w,p) where t is an index term, w is its associated weight, and p is a pointer to the next document containing t. Only the weight is an optional component; the term identifier is necessary to detect matches and to differentiate among the chains which intersect in a document. Because a pointer is required for each assigned index term, the total



Legend

t=index term
w=weight
c=correlation

D=document vector
R=retrieval data
a=document accession number

Structure and Search of a Document File
Using a Chained Organization
Figure II-1

space needed for them is considerable. Depending on the number of bits allocated to each element of the triple, the file size may increase as much as 100% (12). The space required for the file directory is small in most cases and presents no genuine implementation problem.

In order to alleviate some of the overhead for pointers, the multi-list file combines several descriptors into a "superkey" and maintains a single chain for items containing all descriptors (13, 14). To effect a savings, the vocabulary must contain term pairs or triples which occur a significant number of times throughout the file. Unfortunately there are very few super-keys which meet these conditions. Tests involving triplets show that 90% of the super-keys occur only once. As a result, the multi-list structure is only slightly better than chaining single keys and does not provide the storage economy that is needed.

Information systems with a variety of interrelated record types often use chains to reduce the storage of redundant information as well as to provide the desired access points. The Integrated Data Store (15) circularly chains common data elements and addresses them by pointers from other parts of the file. Similarly, records having the same attribute values are chained in rings which may also point to other rings. Logically, the file is a highly inter-connected network of data elements and records accessed through a single directory. Physically, items are packed into pages and fetched from disk through the data management facilities of COBOL. Several other programming systems, CLP (16), APL (17), and DMS (45) provide facilities for defining and manipulating similar structures. Although ideal for many files, ring structures provide few benefits for information retrieval since the typical document file has only one record format, little repetitious data, and few complex linkages.

Considering data retrieval in chained organizations, the search time is proportional to the total length of the chains involved plus a small amount for the directory scan. Naturally chains intersect at various places, and duplicate effort may result. For example, if one chain is followed to its end before another is considered, then some documents are visited twice and unnecessary accesses are made. This situation is eliminated by considering the next element in all chains before fetching a new document vector. Since each chain is ordered by accession number, always selecting the highest number insures that no items are re-scanned. In addition, all correlations are gathered in a single sweep across the disk, i.e. without jumping back across data previously passed over. If only recent data are required, it may be possible to terminate the search early. Specifically, the accession numbers used as pointers might contain an indication of publication date or dates could be examined directly in the vector. Regardless of these factors, search time is still approximately one access per document correlation, although this may be reduced somewhat further by using blocked records. Although better than sequential files, it is doubtful that a chained organization provides access that is fast enough for on-line operation.

Lefkovitz (1) describes two variations of the chained organization which are designed to reduce search time--the controlled list length and cellular organizations. In the first case, each document chain is limited to a pre-determined maximum size. When the maximum is exceeded, another directory entry is made and the list is continued. In the second case, a new directory entry is made each time the chain crosses a cell boundary. A cell is defined as a convenient sub-unit of the storage device; for disks, the cylinder is an appropriate choice. In either case, the directory size and scan time increases. In systems testing for complete matches of query terms, it is possible to intersect sections of chains and thereby eliminate some documents not having all query terms. However, matching based on correlation functions allows documents to be retrieved even if they do not completely match the query. As a result, all chained items must be examined and the intersection process is of little value.

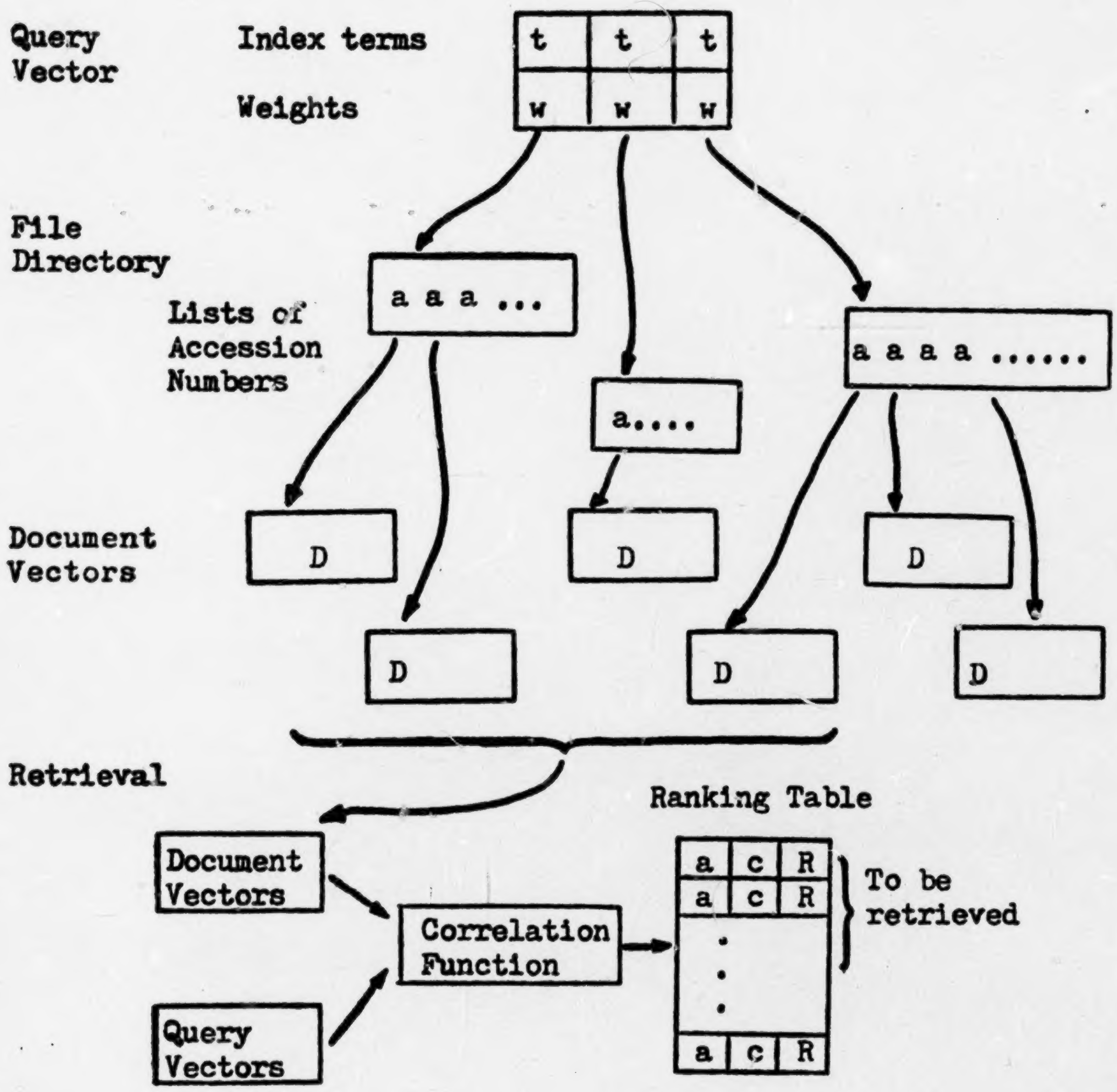
Chained file organizations have some advantages in an on-line system. Pointers make it possible to relate documents in a wide variety of ways; e.g. similarity of keyword assignments or bibliographic data; statistical correlation; or _a priori_ knowledge of related publications, former editions, or other factors. Moreover, a number of pre-search statistics are available to aid query formulation. For example, chain length indicates the number of items indexed by each term (specificity of the query) and the total list length is certainly an upper bound on the number of retrievable documents. Such information might form a useful display. Real-time updating is also possible since only a few pointers must be changed to incorporate new items into the file. Finally, since document vectors are stored intact, both relevance feedback and space modification procedures are possible.

Nevertheless, _the disadvantages of chains outweigh their advantages_. First, precautions must be taken to avoid and repair breaks due to hardware or software failure. Second, in spite of easy file addition procedures, changing an existing document is difficult, especially if its vector must be enlarged. Finally, the main objections are the heavy storage overhead for pointers and search times proportional to the number of documents correlations. The last disadvantage is a particular hindrance since better retrieval is generally obtained by examining an increasing number of documents.

C. Inverted Files

The inverted file organization has wide use in on-line retrieval systems because it provides quick access to data (18, 19, 20, 21). Logically the file is partitioned into sets of documents having a common keyword, although records themselves are stored in any order. The directory contains one entry per vocabulary term; each entry lists the accession numbers of documents containing that term. In a sense, documents are chained, but all pointers reside in the directory rather than in vectors. Figure II-2 depicts the structure and use of a file organized in this way, that is, using lists of accession numbers inverted by index term. A search begins by scanning the directory to obtain the lists associated with each query term. After merging the lists, each vector is fetched from disk, matched with the query, and the resulting correlation stored in a ranking table. Finally, the highest scoring documents are retrieved and displayed.

As outlined, the inverted file produces no better performance than a chained organization. The retrieval time is still proportional to the number of correlations since complete vectors are fetched from random



Legend
a = document accession number
c = correlation
t = index term
w = weight
R = retrieval data
D = document vector

Structure and Search of a Document File Using Lists of Accession Numbers Inverted by Index Term

Figure II-2

disk locations. In fact, the situation is worse since the directory is quite large and cumbersome. However, retrieval time can be substantially reduced by eliminating many items during the directory scan. For systems using Boolean queries, this is clearly the case since conjunction of search keys imply list intersections while disjunctions imply unions. As a result the only documents actually fetched are the ones which completely match the request.

A similar procedure is possible with some correlation coefficients for unweighted vectors. The ranking table is used as an area for accumulating the number of matching terms between the query and various documents. For many match functions, these totals constitute the numerators of the coefficient; normalization is all that is necessary before the final value is obtained. There are several ways to obtain the normalizing denominators for coefficients:

1) access their document vectors,
2) provide a special table, or
3) include them with the accession numbers in the directory lists.

None is very pleasing. The first case returns to the situation of accessing all items on the merged list in order to obtain a complete set of correlations. The other methods are faster, but increase the directory size. Regardless of cost, it is possible to obtain complete correlations after just the directory scan, and the resulting savings are considerable. Retrieval time is reduced to an amount proportional to the number of query terms plus the time needed to fetch data for final display--certainly fast enough for on-line work. Moreover, since correlations are computed from directory information, index terms are no longer needed as part of the stored document vectors. In fact, if only accession numbers are output, the entire file might consist of just the directory, resulting in storage requirements approximately the same as for a serial file. This is certainly a pleasing situation for initial searches, but it obviates the use of relevance feedback and document space modification.

To review, both relevance feedback and space modification produce increased user satisfaction and are, therefore, desirable tools in a retrieval environment. However, they require the entire contents of document vectors--in one case to obtain index terms and in the other case to modify them. The inverted directory contains all this data but in term order rather than document order. Consequently the only realistic way to implement these features and maintain on-line operation is to have both the inverted and main files present. This combined file approach nearly doubles the storage requirement. And in the case of space modification changes must be made to both files.

The inverted organization can be modified to work with weighted query and document vectors also. On one hand, the system could operate as outlined in Figure II-2. On the other hand, the directory can be augmented to permit a more rapid search. For example, let $D = (d_1, d_2, \ldots, d_n)$ and $Q = (q_1, q_2, \ldots, q_n)$ be a document and query vector respectively. Here d_i and q_i denote the weights assigned to the i^{th} term. The cosine correlation between these vectors is:

$$\text{COS}(Q,D) = \frac{Q \cdot D}{|Q|\ |D|} = \frac{\sum_{i=1}^{n} \left[q_i\ d_i \right]}{\left[\sum_{i=1}^{n} q_i^2 \right]^{\frac{1}{2}} \left[\sum_{i=1}^{n} q_i^2 \right]^{\frac{1}{2}}} \qquad \text{(II-1)}$$

In order to compute COS(Q,D) during the directory scan, the values of $d_i/|D|$ are stored in the appropriate directory lists along with accession numbers. This is possible since all the needed information is available when the document is added to the file. During a search, the total correlation is accumulated in the ranking table by summing the contributions from matching terms. The contribution from matching terms with weights q_i and d_i is:

$$\text{CONTRIBUTION} = \frac{q_i d_i}{|Q|\,|D|} = \left(\frac{q_i}{|Q|}\right) \cdot \left(\frac{d_i}{|D|}\right) \qquad \text{(II-2)}$$

The lest hand factor in the last equality is obtained from the query and the right hand factor from the directory. It is important to note that normalization is already included in the directory entries. An example of correlation calculation is shown in Figure II-3. With this procedure all correlations are obtained after the directory scan and only the retrieval data for the highest scoring documents must be fetched. Further, both terms and weights may be removed from the original vectors if feedback and space modification procedures are not used. Assuming the system design includes these features, a combined file must be maintained; again the storage overhead is approximately 100%.

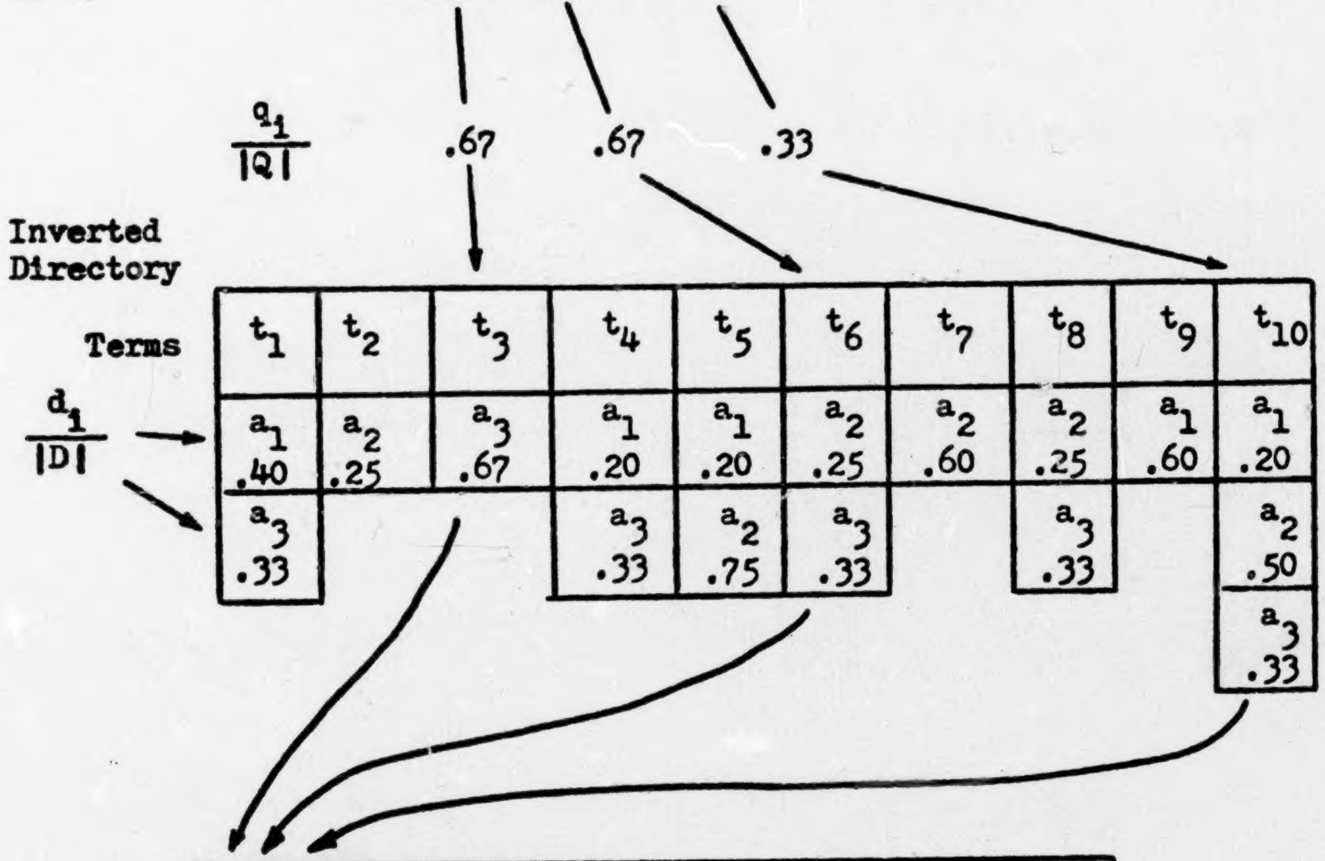
The advantage of computing a correlation during the directory scan is the rapidity of the search. Although this technique works with the cosine correlation, it is inconvenient or impossible with others. Each

Documents $D_1 = (2,0,0,1,1,0,3,0,3,1)$ $|D_1| = 5$

$D_2 = (0,1,0,0,3,1,0,1,0,2)$ $|D_2| = 4$

$D_3 = (1,0,2,1,0,1,0,1,0,1)$ $|D_3| = 3$

Query $Q = (0,0,2,0,0,2,0,0,0,1)$ $|Q| = 3$


$\frac{q_i}{|Q|}$
.67
.67
.33
Inverted Directory
Terms
$\frac{d_i}{|D|}$
t_1 a_1 .40 a_3 .33
t_2 a_2 .25
t_3 a_3 .67
t_4 a_1 .20 a_3 .33
t_5 a_1 .20 a_2 .75
t_6 a_2 .25 a_3 .33
t_7 a_2 .60
t_8 a_2 .25 a_3 .33
t_9 a_1 .60
t_{10} a_1 .20 a_2 .50 a_3 .33

Ranking Table

Doc.	Correlation
a_1	$(.33)(.20) = .06$
a_2	$(.67)(.25)+(.33)(.50) = .33$
a_3	$(.67)(.67)+(.67)(.33)+(.33)(.33) = .78$

Legend a_i = accession number for D_i Q = query vector

D_i = document vector i t_i = term i

Sample Correlation Calculation Using an Inverted File with Contributions Stored in the Directory Lists

Figure II-3

matching function has its own peculiarities, but in general two requirements must be met. First, non-matching document terms must not have a direct influence on the coefficient since only the lists corresponding to matching terms are examined. However, in some cases, such information may be derived from other quantities already on hand. Second, the computation must permit the accumulation of total coefficient. In the case of the cosine function, this is accomplished by storing normalized contribution values in the lists of accession numbers. Of course, extra storage allocated to each list element could facilitate almost any computation, but practical considerations generally impose some limitations. As examples, consider the product-moment, Tanimoto, and overlap correlations:

$$PM = \frac{(Q-\bar{Q}) \cdot (D-\bar{D})}{|Q-\bar{Q}| \; |D-\bar{D}|} \qquad \text{(II-3)}$$

$$T = \frac{Q \cdot D}{n\bar{Q} + n\bar{D} - Q \cdot D} \qquad \text{(II-4)}$$

$$O = \sum_{i=1}^{n} \frac{\text{Min}\,(q_i, d_i)}{\text{Min}\,(n\bar{Q}, n\bar{D})} \qquad \text{(II-5)}$$

where $\bar{Q} = \frac{1}{n} \sum q_i$ and $\bar{D} = \frac{1}{n} \sum d_i$. The product-moment measure cannot be computed in the manner suggested because of negative contributions from non-matching terms. The Tanimoto and overlap coefficients do not allow for easy calculation since values of q_i, d_i, $\sum q_i$, $\sum d_i$ must be available. In order to compute either correlation, weights for document terms must be included in the accession lists and the values of $\sum d_i$ must be obtained from a separate table.

Henceforth, it is assumed that system design permits calculation of correlations within the directory so that the problem of quick response reduces to the question of rapid access. Unfortunately accession lists have variable and often considerable lengths and many standard directory scanning techniques do not work. Length is especially troublesome if automatic document indexing is used and nearly every major word is treated as an index term. Without other controls, it is not uncommon for a list to span several disk tracks. Collmeyer and Shemer (22) consider forming serial, tree-structured, and hash-coded indexes to the accession lists and conclude that hashing is preferred. With a hashing procedure, the storage location of an accession list is computed from the bit pattern for the corresponding index term. Higgins and Smith (23, 24) give considerable attention to hash-coded indexes, looking at methods for computing addresses and handling overflows. Equal attention is given to the storage of accession lists to permit easy access and maintenance. Exponential chaining is developed as technique for handling file additions. Generally, in the updating process accession lists must be lengthened, possibly by re-writing them or by chaining the overflow to the original list. In time, the directory deteriorates into chains of updated entries, which slows processing. The exponential chaining procedure adds an entire block of available space to the overflow chain each time the previous block is filled. By continually increasing the size of the additional block, lists are kept from being too fragmented in storage. A variation of this scheme uses periodic maintenance runs to collect all segments of a chain and to re-write it as a single block along with an exponentially increasing number of available overflow spaces.

To summarize, the inverted file search examines only pertinent records and yields an acceptable search time at the price of increased storage space. The time results from three factors:

1) index searching to locate the appropriate accession lists,
2) fetching accession lists and computing correlations, and
3) obtaining retrieval data for the documents to be displayed.

The first two factors are proportional to the number of query terms while the third is related to the amount of desired output. It is unfortunate that search time increases with query length since this penalizes longer queries which generally perform better. In addition, since the relevance feedback process adds additional terms to queries, the second and third iterated searches are considerably more expensive. The storage space and complexity is the big drawback to this organization. In order to get all the desired features, combined files must be used and substantial penalties incurred in terms of space and maintenance. The problems of long accession lists, overflows, and chained blocks within lists are considerable. Generally, the same retrieval features available with the chained organization are also applicable to the inverted file.

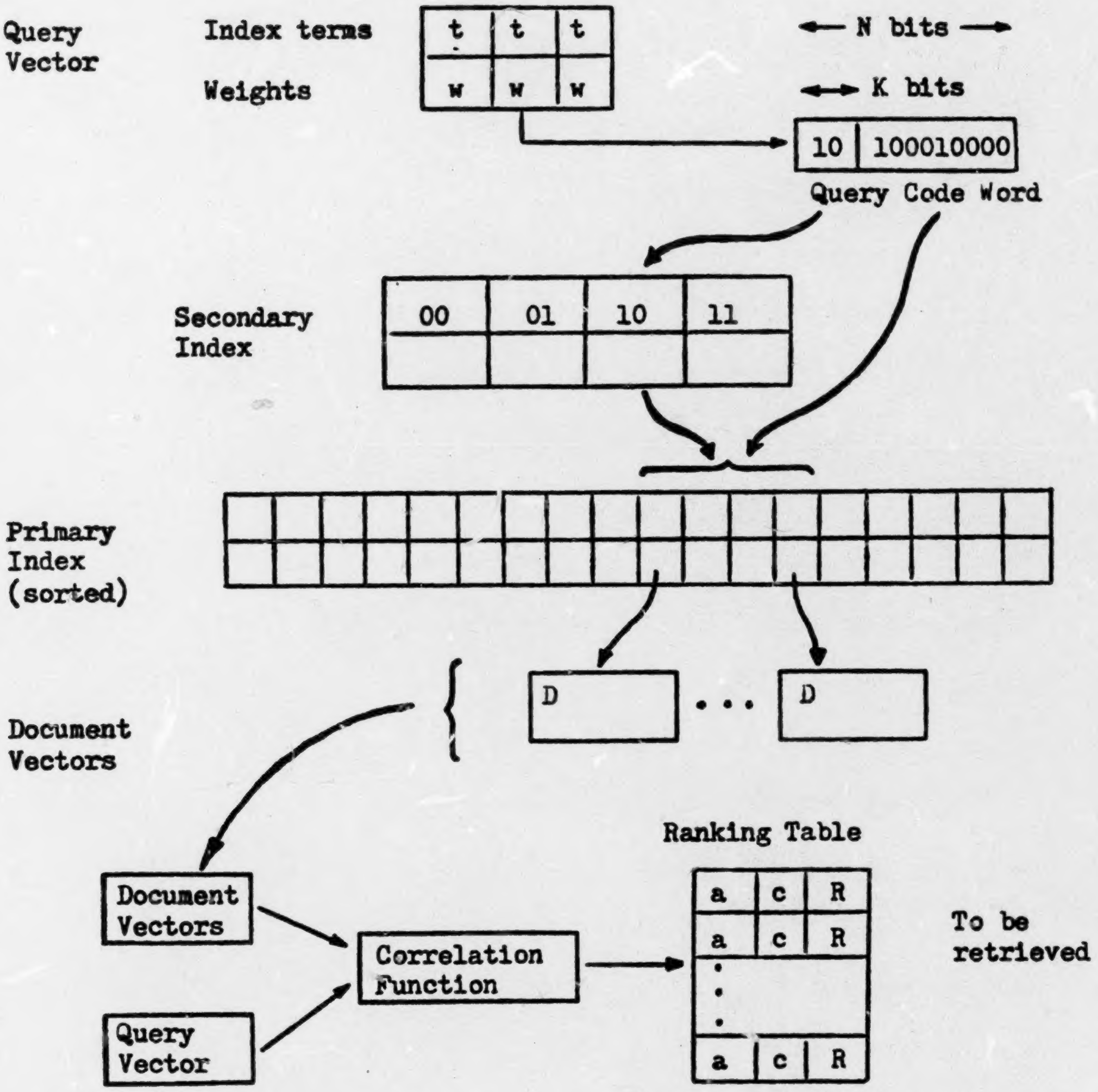
D. Computed-access Files.

Computed-access files are those that try to approach large scale content addressable memory. In other words they manipulate the contents of a query vector to calculate the storage address of a group (bucket) of documents which are highly similar to the query. Hopefully, many of these documents are actually relevant. Hash addressing (scatter storage) is a computed access method that has been used successfully with a single search key (25, 26, 27, 28, 29). To extend this idea to several keys (index terms), a conglomerate address must be calculated which is descriptive of the entire query or document and the query addresses must be mapped into the document addresses. The first scheme discussed below follows this idea to some extent; while the second scheme uses concepts from finite geometries to compute addresses. Regardless of the method, calculated-access differs from previous organizations in that:

1) the file is partitioned into groups of items related in some mathematical manner and
2) a major portion of the access procedure is based on computation rather than on a directory scan.

Files and Huskey describe a retrieval system using super-imposed coding which partially meets the above criteria (30). Document vectors are maintained in serial order as described in Section II.2A. A directory entry is made for each document consisting of a code word (N bits) and a pointer to its vector. This directory is kept in serial order also. Now consider a specific document. The character string for each of its index terms is hashed to a value between I and N, and the corresponding bit in the document code word is turned "on". All other bits remain "off". Once the document code word is generated in this manner it is placed in the directory. To search the file, a query code word is generated by the same hash function and is matched with all document codes. Each time the query code is a subset of a document code, the pointer to the document is saved. Later the pointers are used to access complete vectors and correlations are computed (See Figure II-4).

The scheme appears neat and simple, but depends on making the number of bits per code word large enough to eliminate too many false drops, and at the same time small enough to make the directory manageable. The serial directory scan is slow, perhaps, but does allow for easy update. For on-line work, more speed could be obtained by sorting the documents by code word. This facilitates bulk transfers of data rather than individual records. Further, if the directory itself is ordered according to the first K bits of its code words, a secondary table might act as an index to it. Given a query code word, the first K bits become a subscript to the secondary index which leads to a substantial section of the directory and then to blocks of documents. As supplemented, this approach might prove feasible with respect to storage space and access time. There is a problem with accuracy, however. Examining a document only when its code word is a complete superset of the query code word may prove too strict. On one hand, documents containing only a few query terms are not examined under this principle, whereas they might actually have high correlations with the query and even be retrieved with a different method. However, examining a document when there is one or more matching code word bits would be such a lenient criterion that most of the file would be scanned. Solutions to these problems approach the clustering techniques and profile generation methods reserved for later discussion. In its original form, the super-imposed coding system is probably not feasible for on-line document retrieval. However, proper



Legend
a = document accession number
c = correlation
t = index term
w = weight

R = retrieval data
D = document vector

File Organization Based on Query and Document Code Words

Figure II-4

enhancements could make it more useful.

For some time, Ghosh and others (31, 32, 33, 34, 35) have considered the following problem. Assume that document vectors are stored serially and consider a directory table holding only the document accession numbers. The problem is to organize the index into buckets so that given a query, the proper bucket addresses are generated by solving algebraic equations. In the context of the work, the "proper bucket" is a bucket containing the accession numbers of documents having all the query terms. The problem is solvable for some special cases: for binary vectors, for queries with a small number of terms, and for queries with two multiple-valued attributes. In general, the organization represents query attributes (terms) by dependent or independent linear forms (hence the bounded query size). The system of equations to be solved in connection with retrieval is $Hx = v$ where H is a matrix of coefficients, x is the vector of query attributes, and v is the vector of attribute values. The elements of H are elements of a Galois field of finite dimensions, sometimes being powers of its primitive elements. The solution $x = H^{-1}v$ is used to generate bucket addresses and the proper documents are obtained.

As fine as the concept sounds, the organization is more a mathematical object than a working reality. The reasons are twofold. First, there are difficulties with the size and computations for the matrix H as well as reducing $Hx = v$ to echelon form, obtaining a solution, etc. Second, the redundant storage factor for these schemes is very high because of the nature of the task, namely computing bucket labels for any combination of terms. Only a few example calculations are available;

but for 3 attributes of 9 values each (729 record types), the scheme uses 81 buckets and, on the average, stores a record 2.8 times. For 2 attributes of 9 values and 3 attributes of 3 values, the redundancy factor is 7.0. As a result, this approach falls short of the desired situation--economically handling thousands of attributes of many values apiece.

Unfortunately, no really good calculated access methods are known at this time.

E. Clustered Files

A clustered file partitions documents into subject classes and uses a hierarchy of profiles to describe and access each cluster. The term clustering refers to the use of a classification scheme to produce groups (clusters) of statistically related items. Early research on classification methods was conducted by Needham (36), Doyle (37), Parker-Rhodes (38), and others. The combination of document clustering, profile hierarchies, and on-line searching developed more recently, mainly due to the work of Salton's SMART project (39, 40, 41, 42). Because a great deal of descriptive material is presented in Chapter III, the discussion here is confined to relating clustered files to other organizations.

Logically, the file is partitioned into subfiles (clusters) using the similarity criteria of the classification procedure. Generally, many of the grouped documents share several index terms rather than just one or two. Hopefully, this implies some semantic relationship among the texts also. However for retrieval purposes, all that is required is for documents in a cluster to have greater internal similarities than external similarities. If profiles are made which reflect this bias, then the best classes for searching are those most similar to the query. As mentioned earlier, a profile is a composite of subordinate vectors, so its format makes it possible to measure query-cluster similarities by profile correlations. The search process begins by computing query-profile correlations for the nodes on the highest level. After ranking, nodes having correlations above a chosen cutoff are expanded. That is, the sons of the nodes are obtained and the correlation-expansion procedure is applied to them. The search, then, is an alternating series of correlations and expansions, terminating with document correlations and actual retrieval.

Throughout the organization, items are stored serially. This includes profiles on the first tree level, the sons of each node, and the clusters of documents vectors. Since large blocks of data (sons of a node) are transferred from disk rather than individual items, the number of accesses made in a search is proportional to the number of expanded nodes. For a very narrow strategy--expanding only the best node at each step--the number of accesses is approximately equal to the number of levels. For broader strategies, the cost increases accordingly. This situation produces a welcome cost-performance tradeoff. A narrow search yields only a few relevant documents, but at low cost; broader searches are more comprehensive and more expensive. Of course, a full file search is always possible by scanning all documents directly. In all cases, the marginal cost of retrieving a few extra documents is small.

The storage overhead for this organization is basically the space for profile vectors. It is difficult to state the total requirement accurately because it depends on the properties of the hierarchy--

number of levels, a degree of nodes within those levels, and the amount of overlap. (See Chapters III and V.) This research shows that acceptable performance levels are maintained even when the profiles are reduced to only 7% of the space allocated to the document vectors. The result is substantiated under a variety of conditions and using hierarchies with widely different properties.

The expense of classification is the largest disadvantage of the clustered file. The number of operations to classify N items may be proportional to N^2, to NlogN or to N depending on the method. There is some evidence that more expensive methods produce better clusters, but cheaper schemes are not completely unacceptable either. This study disregards the classification cost and suggests that hierarchies can be used for several purposes. By distributing expenses among several applications, clustered files are more easily justified. Alternate uses include automated browsing for viewing how information is structured, checkpoint displays during searching, construction of a retrieval thesaurus, query alteration procedures, and others. (See Section 8 of Chapter III.)

The maintenance of a clustered file is a drawback also. For a time new documents can be successfully blended into the existing hierarchy with or without changes to profiles. Eventually the quality of the hierarchy diminishes because the profiles try to represent too much information. This may imply a complete re-clustering. However, a more reasonable scheme is to record the number of additions to each path of the tree and to re-cluster selectively. That is, to re-structure only a single subtree of data and to re-connect it to the rest of the hierarchy.

There are a number of other advantages of a clustered file in addition to those of time and space. Since complete document vectors are available, relevance feedback is easily implemented. Document space modification may require changes to both documents and profiles, however. In addition, almost any correlation coefficient can be computed because complete information is at hand. Several modes of operation might be provided by simply changing the matching function. Cost performance trade-offs and browsing features have been mentioned. The general appropriateness of the hierarchical structure is not to be overlooked either. Its concept is familiar to most users through their previous library activities, and it provides a reasonable structure for browsing and observing the relationships among the stored texts. In summary, the clustered organization not only facilitates storage and access, but also participates in other phases of the retrieval process.

3. Methods of Physical Access

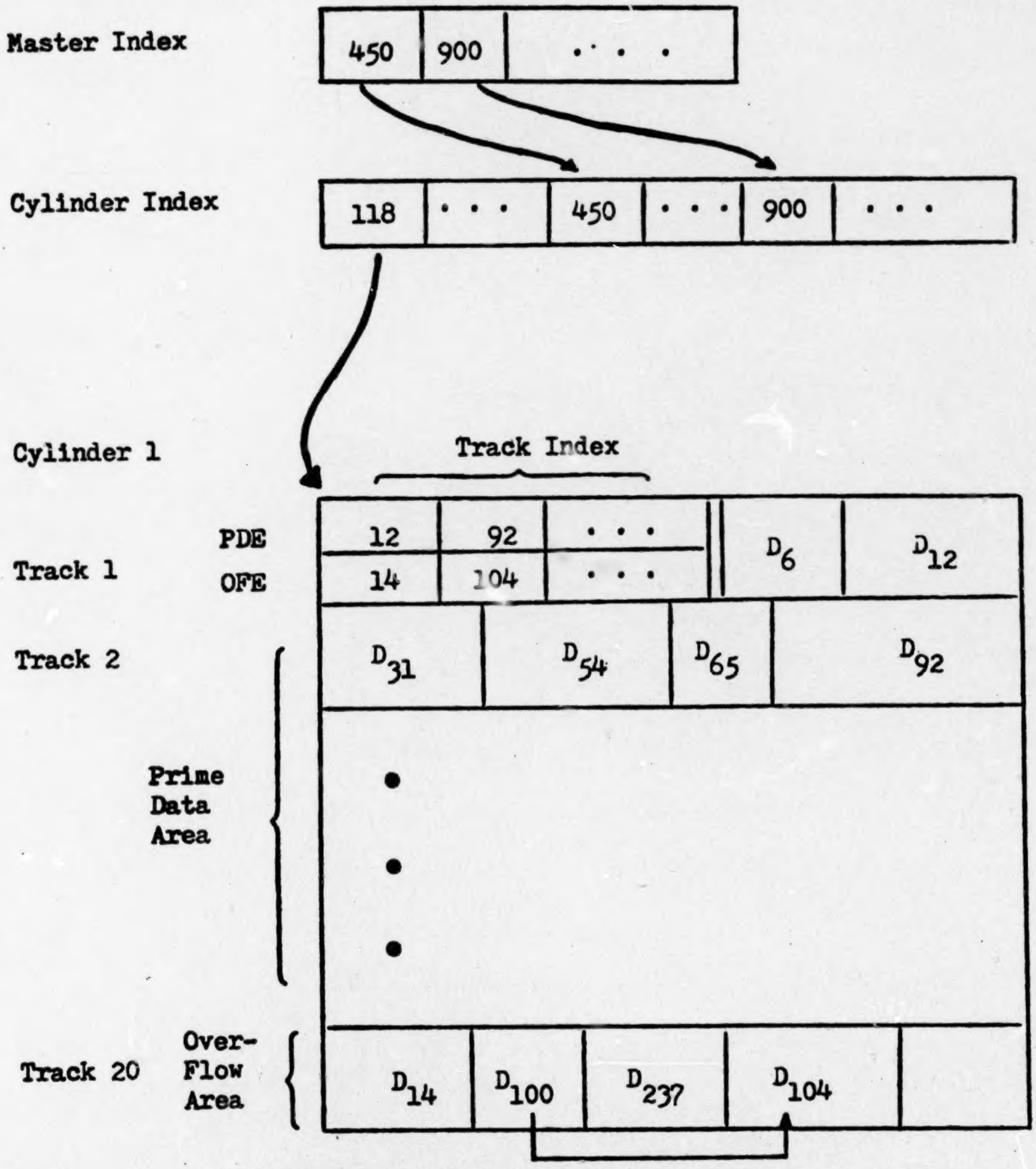
Physical access deals with the process of locating records on a storage device regardless of their relationship with other file items. For example, suppose that the search process follows a chain or expands a cluster and consequently requires correlations with documents D_{17}, D_{48}, and D_{695}. These records must be fetched from disk, deblocked from their track format, and transferred to work areas for the correlation program. The physical access method specifies how the operating system actually finds these vectors on disk. Methods are generally limited to those supported by a manufacturer's software since the task is one of interfacing the operating system and a storage device. Three schemes in wide use are: sequential, direct, and indexed sequential (43).

Sequential access is most applicable to magnetic tape, although used with disk also. It is simple in concept and places little burden on the operating system or application programs. Records are stored in order of increasing reference numbers and located by their relative position in the file. Given the current position, a desired record is found by computing the number of intervening items and reading or skipping over this intermediate information. Although slow for accessing random disk locations, sequential schemes provide the most rapid way of obtaining multiple or quantity input from a localized area of the file.

Direct access is the fastest way to obtain records from random disk locations. The operating system is given the disk address of desired items and actually does little more than initiate I/O activity and transfer input data to program buffers. The application program has the burden of supplying the required disk addresses, generally using lookup or hashing methods. In the first case, the file construction and update processes return addresses of stored items. These are used directly as pointers, elements of accession lists, etc. Whenever an item is needed, its disk address is obtained by table lookup or scanning a list. In the second case, a record accession number is hashed into a disk address and the record is stored at this address or in a connected overflow area. The advantage of hashing is that it avoids a lengthy lookup process; its disadvantages include problems of overflow and selection of a hash function.

Indexed sequential access is easier to use and slower than the direct method, but more restrictive and faster than the sequential scheme. Briefly, the operating system constructs a set of hierarchical directories as the file is built (43, 44, 46). Application programs access items simply by specifying their reference numbers (keys). To start, records are stored in order of increasing key value in the prime data area of each disk cylinder. The rest of the cylinder consists of a track index and an overflow area for new items. During this process, the key for the last record associated with each track is stored in the track index. Moreover, the entire file is preceded by a cylinder index containing the key of the last record associated with each cylinder. Several levels of master index may be used to point to sections of the cylinder index. The result is a hierarchical directory structure which is distributed about the disk (See Figure II-5).

To retrieve data, the operating system (or data management system) compares the key of the desired record with the entries in the highest level index. Descent to lower level indexes is made depending on whether the key has a high, equal, or low match. Prime data tracks are searched sequentially to find the proper record. Actually, retrieval is more complex because of overflow conditions resulting from file updates. Suppose a new record is to be inserted in the middle of the file, but will not fit onto the desired track. In this situation, the entire track is re-written in proper sequence and items pushed off its end are moved to an overflow area. Overflow items are linked together and to their home track by making a second entry in the track index. Consequently when the track index is examined during retrieval, either the prime area is searched sequentially or the overflow area is searched by following a chain. Overflow tracks might be allocated for each cylinder,



Legend: D_x - record for document x

PDE - prime data entries in the track index

OFE - overflow entries in the track index

Index entries are the largest document numbers per cylinder, track, overflow chain, etc.

Indexed Sequential Access

Figure II-5

for the entire file (single area at the end), or both. In the last case, the secondary area contains items overflowing from the cylinder areas.

Indexed sequential access is selected in this research as the method most appropriate to the clustered file. It is particularly useful since the sequence of cluster access is unpredictable and its indexes provide the required retrieval speed. However once a cluster is selected, its volume of data is retrieved as a single entity and sequential, rapid transfer is most important. Because the most efficient operation occurs when entire tracks of information are transmitted from disk, it is assumed that records are not deblocked by the operating system. Master and cylinder indexes are also assumed to reside in core storage as well as the track index for the cylinder currently being accessed. These same conditions are used when the inverted and clustered organizations are compared on the basis of search time. Because direct access may be more appropriate for inverted files, the results are presented in this way also (See Chapter IX).

4. Summary

This chapter surveys current file organizations applicable to direct access devices. In accordance with the purpose of this work, logical structure--partitions of records, linkages among related items--have been emphasized rather than data management functions (physical structure). In all, five types of logical organization are described and evaluated with respect to usage in on-line document retrieval. Table II-1 summarizes the evaluation of sequential, chained, inverted, calculated-

Property	Sequential	Chained	Inverted (Combined)	Calculated Access	Clustered
Partitioning Criteria	None	Single common term	Single common term	Mathematical relation	Correlation many common terms
Linkage To Related Items	None	Pointers	Lists	Computation	Physical adjacency
Space Overhead	None	100%	100%	?	10%
Search Time	Read entire file	Proportional to number of document correlations	Proportional to number of query terms	?	Proportional to number of clusters expanded
Relevance Feedback	Possible, not probable	Possible	Possible with combined file	Possible	Easy
Space Modification	Possible	Possible	Possible but with difficulty	?	Possible but with difficulty
Pre-search Statistics	None	Chain length	List length	None	Limited

Evaluation Summary For Logical Access Methods

Table II-1

Property	Sequential	Chained	Inverted (Combined)	Calculated Access	Clustered
Cost-Performance Tradeoff	No	No	No	No	Yes
Accuracy (Precision-Recall)	Full search	Full search	Full search	?	Varies with search effort
Ease Of Computing Correlations	Very hard	Very hard	Must be of specific type	Very easy	Very easy
Maintenance Addition	Easy	Easy	Moderately hard	Hard	Hard
Changes	Hard	Hard	Hard	Hard	Hard
Browsing	No	No	No	No	Yes
Other					-Clustering expense -Checkpoint search -Alternate uses

Evaluation Summary For Logical Access Methods

Table II-1 Cont'd

access methods to make a complete comparison.

Some comments must be added to the summary. First, all organizations are designed so that relevance feedback and document space modification could be implemented. For sequential files, this involves expending a great amount of time while it means maintaining a combined file for inverted organizations. For clustered schemes, profiles might have to be changed during space modification. The point is that both features can be made available, but with varying costs. Second, in the area of pre-search statistics, the chained and inverted schemes supply the number of documents containing specified index terms, and other information which might be of interest to an on-line user. Clustered files cannot provide this same data, but offer the possibility of interrupting a search and viewing checkpoint information. Third, regarding quality of output, the chained and inverted files supply the same precision and recall values as a full search (serial scan). Exceptions may arise, however, in the treatment of documents with tied correlations. The quality of a cluster search varies with the expended effort so that a cost-performance tradeoff is possible. However, most relevant documents are retrieved early in the search. It might be possible to achieve a similar tradeoff with an inverted organization by examining the accession lists of only a few query terms. However, the system is really processing a different query under this condition. Finally, the major differences between the inverted and clustered schemes is the tremendous storage overhead of the former and the classification expense of the latter. The maintenance difficulties for both are non-trivial. In the end, the additional flexibility and general appropriateness of the clustered file may be the deciding factors in the choices made by actual systems designers.

References

1. D. Lefkovitz, File Structures for On-Line Systems, Spartan Books, New York, 1969.

2. G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill, Inc., New York, 1968.

3. T. C. Lowe, D. C. Roberts, On-Line Retrieval, Informatics, Inc., Project 4594, Technical Report TR-69-304, 1969.

4. C. T. Meadow, The Analysis of Information Systems, John Wiley & Sons, New York, 1965.

5. National Library of Medicine, Medlars System Manual, Bethesda, Maryland, 1965.

6. National Aeronautics and Space Administration, Guide to Processing and Retrieval of Information at the NASA Science and Technical Information Facility (3 volumes), March 1965.

7. A. K. Kent, Searching Literature Files by Computer, Industrie Chemique Belge, Vol. 33, pp. 879-882, 1968.

8. E. G. Coffman, J. Bruno, On File Structuring for Non-uniform Access Frequencies, Technical Report Series, Number 6, Computing Laboratory, University of Newcastle upon Tyne, England, 1970.

9. B. Lipetz, Influence of File Activity, File Size, and Probability of Successful Retrieval on Efficiency of File Structure, Research Department, Yale University Library, New Haven, Connecticut.

10. F. F. Leimkuhler, A Literature Search and File Organization Model, American Documentation, Vol. 19, No. 2, April 1968.

11. S. P. Ghosh, On the Theory of Consecutive Storage of Relevant Records, IBM Research Publication 13627, IBM Research Laboratory, San Jose, California, June 1970.

12. I. A. Warheit, File Organization of Library Records, Journal of Library Automation, Vol. 2, No. 1, March 1969.

13. N. S. Prywes, et al, The Multilist System, Moore School of Electrical Engineering, University of Pennsylvania, Technical Report No. 1 under Contract NOrr551 (40), 1961.

14. N. S. Prywes, H. J. Gray, The Multilist System for Real-Time Storage and Retrieval, IFIPS Proceedings, pp. 112-116, 1962.

15. Introduction to Integrated Data Store, Publication CPB-11048, Computer Department, General Electric Company, April 1965.

16. R. W. Conway, et al, CLP-The Cornell List Processing Language Manual, Department of Computer Science, Cornell University, October 1965.

17. G. Dodd, R. Beach, L. Rossol, APL-Associative Programming Language Users Manual, Research Publication GMR-622, General Motors Research Laboratory, July 1967.

18. R. K. Summit, Dialog--An Operational On-Line Reference Retrieval System, Proceedings of the ACM National Meeting, Thompson Book Co., Washington, D. C., 1967.

19. IBM System 360 Document Processing System, Program Description and Operations Manual, IBM Corporation, White Plains, New York.

20. Project Intrex Semiannual Activity Report PR-12, MIT, Cambridge, Mass., September 1971.

21. Qwik-trieve Users Manual, Westinghouse Tele-Computer Systems Corporation, April 1970.

22. A. J. Collmeyer, J. E. Shemer, Analysis of Retrieval Performance for Selected File Organization Techniques, Xerox Data Systems, El Segundo, California, 1970.

23. L. D. Higgins, F. J. Smith, Efficient Disk Storage for Interactive Document Retrieval, School of Physics and Applied Mathematics, The Queen's University of Belfast, Belfast, N. Ireland, 1970.

24. L. D. Higgins, F. J. Smith, On-Line Subject Indexing and Retrieval, School of Physics and Applied Mathematics, The Queen's University of Belfast, Belfast, N. Ireland, 1970.

25. R. Morris, Scatter Storage Techniques, CACM, January 1968.

26. D. Murray, A Scatter Storage Scheme for Dictionary Lookups, Journal of Library Automation, Vol. 3, No. 3, September 1970.

27. V. Lum, Yuen, Dodd, Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files, Formatted File Organizations Final Report, Information Services Department, IBM Research Laboratory, San Jose, California, March 1970.

28. E. G. Coffman, Jr., J. Eve, File Structures Using Hashing Functions, CACM, Vol. 13, No. 7, July 1970.

29. B. H. Bloom, Space/Time Trade-offs in Hash Coding With Allowable Errors, CACM, Vol. 13, No. 7, July 1970.

30. J. R. Files, H. D. Husky, An Information Retrieval System Based on Super-imposed Coding, Preceedings FJCC, 1969.

31. C. Abraham, S. Ghosh, D. Ray-Chaudhuri, Filing Schemes Based on Finite Geometries, Information and Control, Vol. 12, No. 2, 1968.

32. C. Abraham, R. Bose, S. Ghosh, File Organization of Records for Multiple Valued Attributes for Multivariate Queries, IBM Research Report 1886, 1967.

33. C. Abraham, R. Bose, S. Ghosh, File Organization of Records with Unequal Valued Attributes for Multi-attribute Queries, preliminary report to Air Force Contract AF 30(602)-4088, May 1967.

34. S. Ghosh, C. Abraham, Application of Finite Geometry in File Organization for Records with Multiple-valued Attributes, IBM Journal of Research and Development, Vol. 12, No. 2, 1968.

35. S. Ghosh, Organization of Records with Unequal Multiple-valued Attributes and Combinatorial Queries of Order 2, Formated File Organization Techniques Final Report, Information Services Department, IBM Research Laboratory, San Jose, California, March 1970.

36. R. Needham, The Place of Automatic Classification in Information Retrieval, Report ML-166, Cambridge Language Research Unit, Cambridge, England, 1963.

37. L. Doyle, Breaking the Cost Barrier in Automatic Classification, SDC paper SP-2516, July 1966.

38. A. F. Parker-Rhodes, R. M. Needham, The Theory of Clumps, Report ML-162, Cambridge Language Research Unit, Cambridge, England, 1960.

39. J. J. Rocchio, Document Retrieval Systems--Optimization and Evaluation, Harvard University Doctoral Thesis, Report ISR-10, Chapter 4, to the National Science Foundation, Harvard Computation Laboratory, March 1966.

40. J. D. Broffitt, H. Morgan, J. Soden, On Some Clustering Techniques for Information Retrieval, Report ISR-11 to the National Science Foundation, Cornell University, June 1966.

41. R. Dattola, A Fast Algoritm for Automatic Classification, Journal of Library Automation, Vol. 2, No. 1, March 1969.

42. R. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

43. IBM System 360 Operating System, Supervisor and Data Management Services, IBM Corporation, White Plains, New York, 1968.

44. V. Lum, H. Ling, M. Senko, Analysis of a Complex Data Management Access Method by Simulation Modeling, Information Services Department, IBM Research Laboratory, San Jose, California, 1970.

45. Data Management System (DMS) Reference Manual for XDS Sigma 5/6/7/9 Computers, Xerox Data Systems, December 1970.

46. A Description of AMIGOS, Comress Corporation, Rockville, Maryland, 1970.

Chapter III

Clustered Files

1. Introduction

The purpose of this chapter is to present a comprehensive discussion of clustered files. Individual sections are devoted to classification methods, hierarchy formation, search techniques, updating, cluster storage, request clustering, and alternate uses for the hierarchy. The intent is to provide an understanding of the construction, use, and maintenance of clustered files and to introduce concepts used in the experiments described in later chapters.

2. Classification Methods

A clustered file organization depends on a classification algorithm to group documents with similar properties. Since document properties (keywords) reflect subject content, clusters actually consist of semantically related items even though the partition is made on a statistical basis. Classification methods can be characterized in at least two ways: by direction of application and by amount of work. Generative methods arrive at the final hierarchy by "bottom-up" processing. Initially all documents are considered as individuals and highly similar items are placed in classes (clusters). A profile is constructed to represent each class and higher level clusters are produced by grouping profiles. The process continues until only a small number of items remain. Divisive methods construct the hierarchy in "top-down" fashion. Initially all documents are placed in a single class which is divided into a few large subclasses. Thereafter each subclass is treated independently and is split into additional subclasses. This division process continues until the size of a subclass falls within specified limits. Regardless of the method, clustering entails considerable expense, and algorithms can be characterized by the number of operations required to classify the N items on a single level. A great many algorithms employ similarity matrices giving pairwise associations among items; in general N^2 operations are required to generate such a matrix. Other algorithms partition or group items into crude subclasses and then refine the division. Often, the elements of each subclass are compared with profiles for other classes and then shifted about. If there are k subclasses, then the total clustering effort is proportional to kN operations. Some schemes do not re-distribute items after assigning them to initial subclasses and thus handle them only once per level. These are so-called one-pass algorithms. Apart from top-down and bottom-up application and the work involved, classification algorithms are based on a wide variety of techniques including eigenvalue analysis, factor analysis, latent class analysis, clump theory, and others (1). For most information retrieval applications, a scheme must provide control over cluster size and overlap, and at the same time it must not demand excessive computation or storage space.

Sokal and Sneath (2) describe a number of clustering algorithms using similarity matrices. For the present, let S_{ij} denote the similarity between documents i and j. In the single linkage method, the similarity between classes A and B is defined as

$$S_{AB} = \text{Max} \left\{ S_{ij} \;\middle|\; i \in A,\ j \in B \right\}$$

The hierarchy is made by choosing a threshold θ and joining pairs of documents or classes which have similarity coefficients greater than θ. The threshold is reduced and pairing resumes; the entire process continues until a termination condition is satisfied. The average weight method is similar, but uses a different measure:

$$S_{AB} = \frac{1}{n_A n_B} \sum_{i \in A} \sum_{j \in B} S_{ij}$$

Here n_A and n_B are the number of items in their respective classes. The complete linkage scheme joins two classes if and only if $S_{ij} > \theta$ for all $i \in A$, $j \in B$. Each of these algorithms generates the hierarchy from the bottom up and requires N^2 calculations to make the similarity matrix plus the work involved in the pairing process. Unfortunately, the problems of matrix storage and calculation generally prohibit the use of N^2 methods in document classification.

One-pass classification algorithms examine each item only once per level. Again, the procedures rely on similarity coefficients S_{ij} and a threshold θ. As records (documents or profiles from the previous level) are read from the input device, the first item starts a cluster. The second item is compared with the first: if $S_{12} \geq \theta$, the second item is assigned to the same cluster; if $S_{12} < \theta$, the second item starts a new cluster. Subsequent items are compared with all previous classes and either join the best existing class or start a new one. Nagy and Casey (3) require each item added to a cluster to have a similarity greater than θ with all previous members. Hill (4) and Johnson and La Fuente (5) avoid examining items in a cluster by using a profile to represent its members. Similarities are measured using the profile and altering its makeup to reflect the addition of new items or the loss of old ones.

The cutoff θ can remain constant or vary to provide better control over the size and number of clusters (5). Obviously, one-pass methods are applicable in top-down or bottom-up fashion. The work involved is difficult to establish because the number of clusters changes as the algorithm proceeds. However if K clusters are eventually formed from N items, the total work for that level is bounded by kN operations. One-pass methods are definitely economical of time and space, but are questionable in terms of quality. Although promising results are known (5), additional research is needed in this area.

Between the N^2 and one-pass methods is a class of top-down algorithms which re-adjust an initial partition of the eocuments in order to improve the cluster quality. Rubin and Friedman (6) start with a random partition and use hill-climbing techniques, "forcing passes", and "re-assignment passes" to minimize the data scatter within clusters. Litofsky (7) partitions the vocabulary of a node (all index terms in the corresponding documents) before associating documents with these sub-vocabularies. Similar methods by Doyle and Dattola (8, 9) are used exclusively in this study. To start, a number of unrelated documents are chosen as cluster seeds and their vectors are used as profiles. Next follows the scoring cycle in which each document is compared with all profiles. If for each document, the highest document-profile score is above a given threshold θ, the document is placed in the corresponding cluster at the end of the cycle. If the highest score is less than θ, the document is assigned to a special class of loose items. After all documents are scored, the cycle ends; profiles are re-defined to reflect the gain or loss of cluster members; and parameters are adjusted to control size and overlap.
When a cycle results in no changes to the classification, an iteration is said to have ended. At this point, if the number of loose documents is

large, the cutoff θ is lowered and a new iteration begins. If only a small number of items remain loose, processing terminates for this level. Remaining loose documents may be blended into existing clusters or passed on to the next hierarchy level as individual items. Although the algorithm appears complex, it has a number of desirable features, including control of cluster size, overlap, and disposition of loose items. The process is applied in top-down direction and works in time proportional to kN operations per level, where k is the number of clusters on that level.

At this time, optimal classification methods have not been found for use in document retrieval. All methods are moderately successful depending on the data and amount of computation, but still not enough is known about methods for automatically identifying and grouping related pieces of text. The difficulties may lie in the document indexing or in the classification methods themselves. Classification is not the topic for investigation in this research; and although its problems are not mentioned further, the limitations of current knowledge does affect the results presented here.

3. Hierarchy Formation

A cluster hierarchy is a connected arrangement of profiles which act as a file directory. In the previous section, cluster generation is described as a level by level process having one set of profiles associated with each level. The hierarchy is constructed by connecting each profile to its constituent elements on the next lower level. The result is a tree structure whose roots are top level profiles, whose leaves are documents, and whose intermediate nodes represent the middle level profiles. The tree is used for both retrieval and updating functions, and in both cases each profile's purpose is to characterize all documents beneath it. For retrieval, the documents most similar to a query are found by comparing the query with the first level profiles and following the most promising paths. For updating, a new document is treated as a query and its vector is stored with the most similar lowest level cluster. In addition to the broad questions concerned with the hierarchy structure, two topics of specific concern are methods of linkage among nodes and methods of defining profiles.

A. General Structure

A systems designer needs to satisfy the desires of several types of users and to maintain a balance among response time, search cost, and retrieval performance (precision-recall). To some degree, these measures depend on the general shape of the hierarchy--cluster size, amount of overlap, and number of levels, all of which are interdependent themselves. Large clusters of documents or profiles produce a hierarchy with only a few nodes and levels compared to a hierarchy based on small clusters. As a result a search consists of fewer profile comparisons and is quicker and cheaper. The output, however, is inferior since presumably it is more difficult to detect the presence of a relevant document by examining the profile of a large cluster than to detect it by examining the profile of a small cluster.

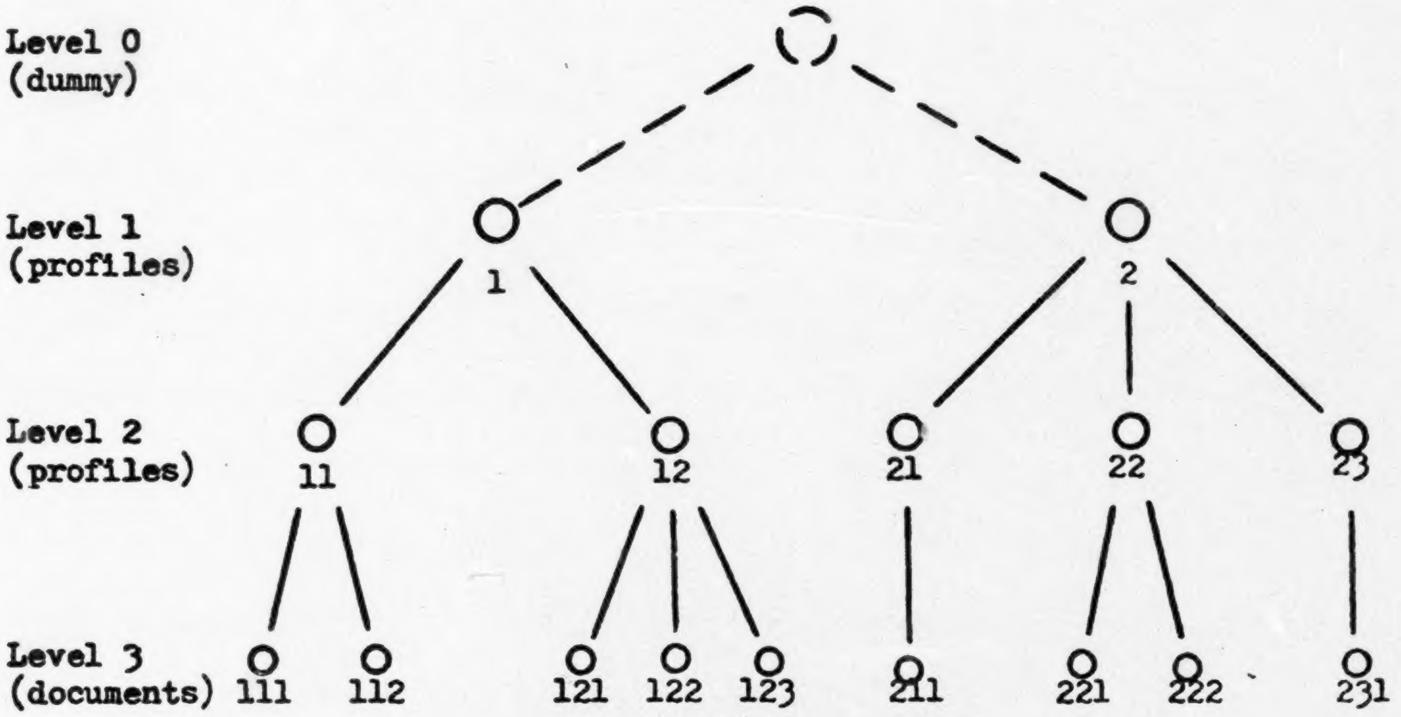
Not a great deal is known about hierarchical shape and most classification algorithms avoid the problem by providing parameters for cluster size, overlap, number of levels, etc. An experimental approach to finding optimal cluster size is to classify a sample collection several times, to search it using the same search strategy and keeping the amount of work constant, and to determine the best cluster size based on the precision-recall plots. Even if this difficult test sequence is maintained, the results may not be applicable to different classifications, search parameters, or collections. The problem of controlling the experiment centers on keeping the search effort constant while changing the cluster size. A more basic difficulty is that of choosing an adequate and appropriate measure of search effort. The above test scheme is difficult, but superior to methods which examine only search time or storage space. At this point, no optimal cluster size is known. The values used in this study are based on the findings of current research and their appropriateness to practical stituations.

In general, clusters overlap so that a document may have multiple memberships, and may be retrieved from several search paths. The optimal amount of overlap has been investigated in a manner similar to that outlined above. That is, cluster size is held constant while overlap is varied and the evaluation is based on precision-recall curves from searches using fixed strategies. The problems related to work measurement are simplified because cluster size is constant. Results by Dattola (9) suggest that a small amount of overlap is best (2%-10%). This is an important finding because of its influence on search times and the method of linkage among nodes of the hierarchy.

B. Linkage Among Nodes

Linkage between levels of profiles or profiles and documents can be accomplished in at least two ways. Pointers might link a parent to its sons either by starting a chain in the parent node and placing one pointer in each son or by collecting all pointers in the parent. With indexed sequential access, record keys are used as pointers rather than disk addresses so that the file remains relocatable. By assigning keys in increments greater than one, new documents or profiles are blended into the file by giving them keys which fill the vacancies in the sequence. Actual vectors are stored in data overflow areas as described in Chapter II. Deleting a record from the file is accomplished by removing its pointers from the hierarchy. Unfortunately, in some cases pointers limit the extent of advanced buffering since the current record must be obtained before the next access is initiated. Another problem occurs if updating overflows the intervals between record keys. However, because pointer linkage is easy to manipulate and is applicable regardless of the amount of overlap, it is often used with a great deal of success.

Implicit linkage methods assign each profile and document a structured key which is a sequence of digits identifying the path locating its node in the hierarchy (Figure III-1a). Given a node for expansion, its key and degree are used to derive the keys of its sons. Buffering may be advanced as far as possible since the keys of all desired records are generated at once. One of the great advantages of a structured key is that it contains a complete description of the record's place in the file. From a single key, parent, son, or filial nodes can be accessed for information that would be useful in broadening or narrowing searches, in browsing processes, or in relevance feedback. In addition, knowledge of the local structure allows for file integrity checks and possible reconstruction following



a) Structured Key Assignments

Order by Subtree	Heir-Filial Order	Order by Level
1	1	1
11	2	2
111	11	11
112	12	12
12	111	21
121	112	22
122	121	23
123	122	111
2	123	112
21	21	121
211	22	122
22	23	123
221	211	211
222	221	221
23	222	222
231	231	231

b) Orders of Cluster Storage

Figure III-1

hardware or software failures. Structured keys meet the requirements of increasing values for indexed sequential access and even permit several storage orders. Right-justifying keys and sorting them into ascending sequence, places nodes in order by levels; left-justifying and sorting results in subtree order; and their combination yields heir-filial order (Figure III-1b). During updating, new pieces can be added to any part of the hierarchy without disturbing previous linkages. As before, the new vector is located in a data or overflow area and only the degree of the parent node is altered. Item removal is best accomplished by setting a deletion indicator and leaving the rest of the record intact. Space is reclaimed during maintenance processing when items are shuffled about to achieve more efficient operation. At this time a record may be assigned a new key, but this poses no serious problems since the keys are only used for internal identification.

Under the assumption of a low percentage of overlap either method of linkage works economically. Structured keys require overlap to be handled by duplicating document vectors and storing them with each cluster in which they have membership. An advantage of duplication is that each cluster becomes a contiguous block of data and therefore accessible at maximum speed. Pointer linkage can be used to save the space given to duplicate items, but cluster access time increases due to the excess jumping about the disk to fetch the overlapping items. The space requirements for a structured key and degree is slightly larger than that needed for pointers and record keys, but probably not enough to make a real difference. Assuming a small amount of overlap, the preference between methods must be given to structured keys, based on the ease of update and the extra links to related information.

C. Profile Definition

One of the assumptions of cluster searching is that profiles for relevant clusters are more similar to a query than other profiles. A similar assumption connected with classification maintains that a high document-profile similarity implies a large degree of similarity between the document and all current cluster elements. Consequently, a good profile definition is crucial to the success of a clustered file. Basically, a profile attempts to characterize all the documents in its crown, that is, all documents reachable from it by descending paths. Given a specific node, a profile is loosely described as a collection of index terms found in those documents. Consider a node whose crown is the document set $C = \left\{D_1, D_2, \ldots, D_n\right\}$. The following standard profile definitions are of general interest:

1) Profile P_1

Let $D_i = (d_{i1}, d_{i2}, \ldots, d_{iv})$ be a vector of unweighted index terms representing the i^{th} document. In particular, let $d_{ij} = 1$ if term j is assigned to D_i and $d_{ij} = 0$ otherwise, for $j = 1, 2, \ldots, v$.

$$P_1 = (p_{11}, p_{12}, \ldots, p_{1v}) = D_1 \vee D_2 \vee \ldots \vee D_n \qquad \text{(III-1)}$$

simply denotes the terms in all clustered items. In other words $p_{1j} = 1$ if and only if there is at least one document containing term j; no importance weighting is used.

2) Profile P_2

Using the same document description, the vector

$$P_2 = (p_{21}, p_{22}, \ldots, p_{2v}) = D_1 + D_2 + \ldots + D_n \quad \text{(III-2)}$$

is a weighted profile in which p_{2j} is the number of clustered items in which term j appears. In this manner, P_2 exhibits document weighting.

3) Profile P_3

Let d_{ij} be the importance (weight) assigned to the j^{th} term in D_i. The profile

$$P_3 = (p_{31}, p_{32}, \ldots, p_{3v}) = D_1 + D_2 + \ldots + D_n \quad \text{(III-3)}$$

exhibits term weighting in the sense that p_{3j} is the total importance assigned to term j in all clustered items.

The definitions begin with two different types of document keywords--weighted and unweighted--and produce either weighted or unweighted profiles. Many retrieval systems using manual indexing do not use weights because keywords are carefully chosen and weighting appears superfluous. In the case of simple automatic indexing, information for computing meaningful weights may not be available. In either instance, P_1 profiles carry out the philosophy of no weighting and produce economical vectors in terms of storage requirements. Although unsophisticated, it should not be surprising to find that P_1 is adequate for distinguishing among dissimilar groups of cohesive documents (clusters). Litofksy uses a modified form of this profile in his work (7).

The P_2 profile starts with unweighted documents and introduces weights to emphasize terms which cause cluster formation. This definition is suitable when converting the unweighted documents of an existing retrieval system to a clustered file organization. Hill (4) and Dattola (9) both use forms of the P_2 vector; in the latter case, rank value weighting is used also.

Automatic indexing generally converts every major word to an index term. As a result, many index terms are assigned to document and it is useful to associate a weight with each term to indicate its importance. In the collections used in this study, term frequencies within a document are automatically assigned as weights. This idea is carried over to P_3 profiles where a weight is the total term frequency in C. The SMART system (10) and the work of Rubin and Friedman (6) make extensive use of P_3 profiles. At this point, it is possible to argue convincingly in favor of each profile definition based on cost-performance, emphasis to inner-document or intra-document properties, or sensitivity to cluster size. Later, investigations consider each definition--its weighting procedures, term deletion, and update characteristics.

In a more theoretical treatment, each document is considered as a point in v-dimensional space. An excellent profile would be the center of mass of the clustered vectors, namely

$$P_{cm} = \frac{1}{n} \sum_{i=1}^{n} D_i \qquad \text{(III-4)}$$

If the values of the vector matching function are independent of the factor $\frac{1}{n}$, then P_2 and P_3 profiles yield the same results as the center of mass for their respective systems. This condition holds for the cosine correlation mentioned earlier, $COS(Q,P) = \frac{Q.P}{|Q|\ |P|}$. However, for the cosine, the centroid vector

$$P_{cv} = \frac{1}{n}\sum_{i=1}^{n} \frac{D_i}{|D_i|} \qquad \text{(III-5)}$$

is a superior profile in the sense that it maximizes the average correlation between itself and all elements of C (11). The difference between P_{cm} and P_{cv} is a matter of a slight shift in emphasis on the terms contributing to the total correlation as evidenced by the following equations.

$$\cos(Q,P_{cm}) = \sum_{i=1}^{n} \frac{D_i}{|D_1 +\ldots+D_n|} \cos(Q,D_i)$$

$$\cos(Q,P_{cv}) = \sum_{i=1}^{n} \frac{1}{\left|\frac{D_1}{|D_1|} +\ldots+ \frac{D_n}{|D_n|}\right|} \cos(Q,D_i) \qquad \text{(III-6)}$$

In the first case, term multipliers are sensitive to vector magnitudes, while in the second case, each term has the same multiplier. Actual differences in values are slight, however, since many documents with similar properties are generally involved in the sums.

The following figures show examples of all profile definitions for node 1 of the hierarchy shown in Figure III-1. Figure III-2 depicts P_1 and P_2 profiles while Figure III-3 shows P_3 vectors. In both cases, the upper half of the drawings show profiles as they might actually be stored on disk. The lower half shows the contributions to the total cosine correlation made by a single match involving each term. As expected, the P_{cm} and P_{cv} vectors produce almost identical contributions. It is also observed that P_3 profiles generally have a larger range of weights than P_2 profiles. As a result, terms of low weights contribute much less to a correlation in P_3 vectors than in P_2 vectors. An opposite relation

Document	Stored Vector	Magnitude
111	$D_1 = (1,0,1,0,0,1,0,0,0,0)$	$\sqrt{3}$
112	$D_2 = (0,0,1,0,0,1,1,0,0,0)$	$\sqrt{3}$
121	$D_3 = (0,0,0,0,0,1,1,0,0,1)$	$\sqrt{3}$
122	$D_4 = (0,0,0,0,0,1,0,0,0,1)$	$\sqrt{2}$
123	$D_5 = (0,0,0,0,0,1,1,0,0,0)$	$\sqrt{2}$
Profiles	$P_1 = (1,0,1,0,0,1,1,0,0,1)$	$\sqrt{5}$
	$P_2 = (1,0,2,0,0,5,3,0,0,2)$	$\sqrt{43}$
	$P_{cm} = (.20,0,.40,0,0,1.0,.60,0,0,.40)$	$\sqrt{1.72}$
	$P_{cv} = (.12,0,.23,0,0,.63,.37,0,0,.26)$	$\sqrt{0.67}$

Contributions of Term Matches to Total Cosine Correlation

Contribution Vector	Matching Term 1	2	3	4	5	6	7	8	9	10
$P_1/\lvert P_1\rvert$	.45	0	.45	0	0	.45	.45	0	0	.45
$P_2/\lvert P_2\rvert$	.15	0	.30	0	0	.76	.46	0	0	.30
$P_{cm}/\lvert P_{cm}\rvert$	.15	0	.31	0	0	.76	.46	0	0	.31
$P_{cv}/\lvert P_{cv}\rvert$	.14	0	.28	0	0	.77	.45	0	0	.32

Legend

$$P_1 = D_1 \vee D_2 \vee D_3 \vee D_4 \vee D_5 \qquad P_2 = D_1 + D_2 + D_3 + D_4 + D_5$$

$$P_{cm} = \frac{1}{5} \sum_{i=1}^{5} D_i \qquad P_{cv} = \frac{1}{5} \sum_{i=1}^{5} \frac{D_i}{|D_i|}$$

Profiles Resulting from Unweighted Document Vectors

Figure III-2

Document	Stored Vector	Magnitude
111	$D_1 = (1,0,3,0,0,1,0,0,0,0)$	$\sqrt{11}$
112	$D_2 = (0,0,2,0,0,3,1,0,0,0)$	$\sqrt{14}$
121	$D_3 = (0,0,0,0,0,1,2,0,0,3)$	$\sqrt{14}$
122	$D_4 = (0,0,0,0,0,3,0,0,0,2)$	$\sqrt{13}$
123	$D_5 = (0,0,0,0,0,2,1,0,0,0)$	$\sqrt{5}$
Profiles	$P_3 = (1,0,5,0,0,10,4,0,0,5)$	$\sqrt{167}$
	$P_{cm} = (.20,0,1.0,0,0,2.0,.80,0,0,1.0)$	$\sqrt{6.68}$
	$P_{cv} = (.06,0,.28,0,0,.62,.25,0,0,.27)$	$\sqrt{0.60}$

Contributions of Term Matches to the Total Cosine Correlation

Contribution Vector	Matching Term 1	2	3	4	5	6	7	8	9	10
$P_3/\lvert P_3\rvert$	.08	0	.39	0	0	.77	.31	0	0	.39
$P_{cm}/\lvert P_{cm}\rvert$	.08	0	.39	0	0	.77	.31	0	0	.39
$P_{cv}/\lvert P_{cv}\rvert$	.08	0	.36	0	0	.80	.32	0	0	.25

Legend

$$P_3 = D_1+D_2+D_3+D_4+D_5$$

$$P_{cm} = \frac{1}{5}\sum_{i=1}^{5} D_i \qquad P_{cv} = \frac{1}{5}\sum_{i=1}^{5} \frac{D_i}{\lvert D_i\rvert}$$

Profiles Resulting from Weighted Document Vectors

Figure III-3

holds in the case of high weighted terms.

As illustrated, P_2 and P_3 profiles assign weights based on document or term frequency. Doyle suggests an alternate profile which characterizes a cluster by a vector of keywords whose weights are rank values (8). A rank value is the difference between a base value and the rank assigned to the term if all terms in the vector are ordered by decreasing frequency. Such a profile is made by the following procedure.

1) Rank all vector terms by frequency; that is, the most frequent term has rank 1, etc. Terms with the same frequency share the same rank.
2) To the i^{th} term, assign the weight $v_i = b-r_i$ where b is a base value (constant) and r_i is the rank given to the term in step 1).

The base value is a pre-selected constant chosen large enough to insure that all v_i are positive in all profiles. Although weights are based on frequencies here, it is easy to see how the rank value technique could be applied to almost any weighting scheme.

Rank value profiles exhibit two major differences from the vectors considered earlier. First, weights are derived from frequency ranks rather than frequency counts, although these two are somewhat related (12). As a result, the weight range in a typical vector is reduced considerably. The reduction is greatest when many documents are considered and the summed frequencies would become quite large. The importance of a small weight range is the reduction in the range of contribution values associated with vector match functions. The second difference is that the constant base value and subtraction process guarantee that all vectors have the same high weight rather than the same low weight. This also decreases the range of contribution values; in fact, the higher the base value, the smaller the range. Figure III-4 shows the application of rank value weighting to the P_2 and P_3 profiles considered previously. Two base values, 5 and 10, are used. Both the use of ranks and the increase in base value are observed to decrease the range of contribution values from those in the previous example.

All forms of profiles mentioned here as well as some variations are the basis for experiments in Chapter V. In particular, P_1, P_2, and P_3 are evaluated under actual search conditions with and without rank value weighting. Each of the major differences of rank value vectors is examined as well as several altogether different weighting procedures. Finally, vector length and internal processing are considered in some detail. This section simply presents the standard and rank value profile definitions and provides several examples of their application.

4. Search Strategies

The search procedure for a clustered document collection can be described as a series of correlations and expansions. Specifically, a query is compared with all nodes on the first level of the hierarchy (correlations) and one or more of the highest scoring nodes are replaced by their sons (expansion). This cycle is repeated until the document level is reached and items are ranked for final output. Narrow searches expand only a few nodes per level and retrieve the most promising documents rather quickly. They provide an inexpensive search with high precision and low recall. Broader searches expand several nodes per

P_2 Profile using Rank Value Weighting		Magnitude
Original Profile	$P_2 = (1,0,2,0,0,5,3,0,0,2)$	$\sqrt{43}$
Frequency Ranks	4 3 1 2 3	
Base Value = 5	$P_r = (1,0,2,0,0,4,3,0,0,2)$	$\sqrt{34}$
Base Value = 10	$P_{r'} = (6,0,7,0,0,9,8,0,0,7)$	$\sqrt{279}$

Contributions of Term Matches to the Total Cosine Correlation

Contribution Vector	Matching Term 1	2	3	4	5	6	7	8	9	10
$P_2/\lvert P_2\rvert$	.15	0	.30	0	0	.76	.46	0	0	.30
$P_r/\lvert P_r\rvert$	.17	0	.34	0	0	.69	.51	0	0	.34
$P_{r'}/\lvert P_{r'}\rvert$	.36	0	.42	0	0	.54	.48	0	0	.42

P_3 Profile using Rank Value Weighting		Magnitude
Original Profile	$P_3 = (1,0,5,0,0,10,4,0,0,5)$	$\sqrt{167}$
Frequency Ranks	4 2 1 3 2	
Base Value = 5	$P_r = (1,0,3,0,0,4,2,0,0,3)$	$\sqrt{39}$
Base Value = 10	$P_{r'} = (6,0,8,0,0,9,7,0,0,8)$	$\sqrt{294}$

Contributions of Term Matches to the Total Cosine Correlation

Contribution Vector	1	2	3	4	5	6	7	8	9	10
$P_3/\lvert P_3\rvert$	.08	0	.39	0	0	.77	.31	0	0	.39
$P_r/\lvert P_r\rvert$	.16	0	.41	0	0	.64	.32	0	0	.48
$P_{r'}/\lvert P_{r'}\rvert$	.35	0	.47	0	0	.53	.41	0	0	.47

Rank Value Weighting Applied to Standard Profiles

Figure III-4

level and secure higher recall, but at lower precision. Thus, by varying the search strategy, it is possible to obtain a desirable cost-performance tradeoff: users wanting a small amount of relevant data have quick, economical searches while users wanting more comprehensive searches pay and wait accordingly.

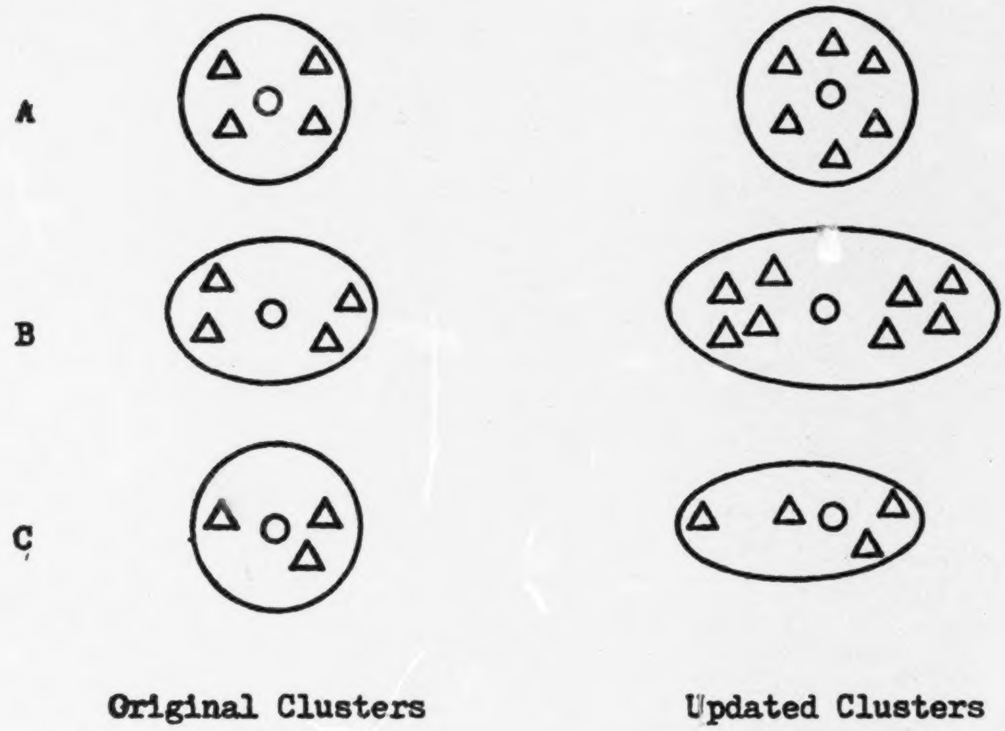
There are a number of variations of this general procedure. Forward search strategies allow only one opportunity to expand nodes on each hierarchy level, thereby constantly proceeding toward the documents. Some systems allow backtracking; that is, restarting the search on a upper level when an earlier expansion turns out poorly. The SMART system, for example, maintains a record of all nodes examined and in each instance expands nodes with the highest correlations. Unless special steps are taken, backtracking occurs in the natural course of events. A plunging strategy is a combination of an extremely narrow forward search followed with backtracking. Initially, only one node on each level is expanded, until the items in the first document cluster are examined and ranked for possible output. Next, a test is applied to determine whether the search should be halted, backed up one or more levels and continued, or completely restarted. The test might involve statistical operations, showing documents to an on-line user, or other criteria. In any case, the search involves narrow plunges to the bottom of the hierarchy. Generally, strategies which include backtracking are worthwhile only if there is a loose, flexible criteria for deciding how many nodes are expanded on each level. For example, it is useless to backtrack if an a priori decision is made to expand a fixed number of clusters. In this case a forward search strategy serves just as well.

Finally, there are a number of problems in measuring query-profile similarity. Consideration must be given to the influence of cluster size, hierarchy level, profile length, and the number and type of matching index terms. The experiments in Chapter V deal with these and other factors since they relate to profile construction.

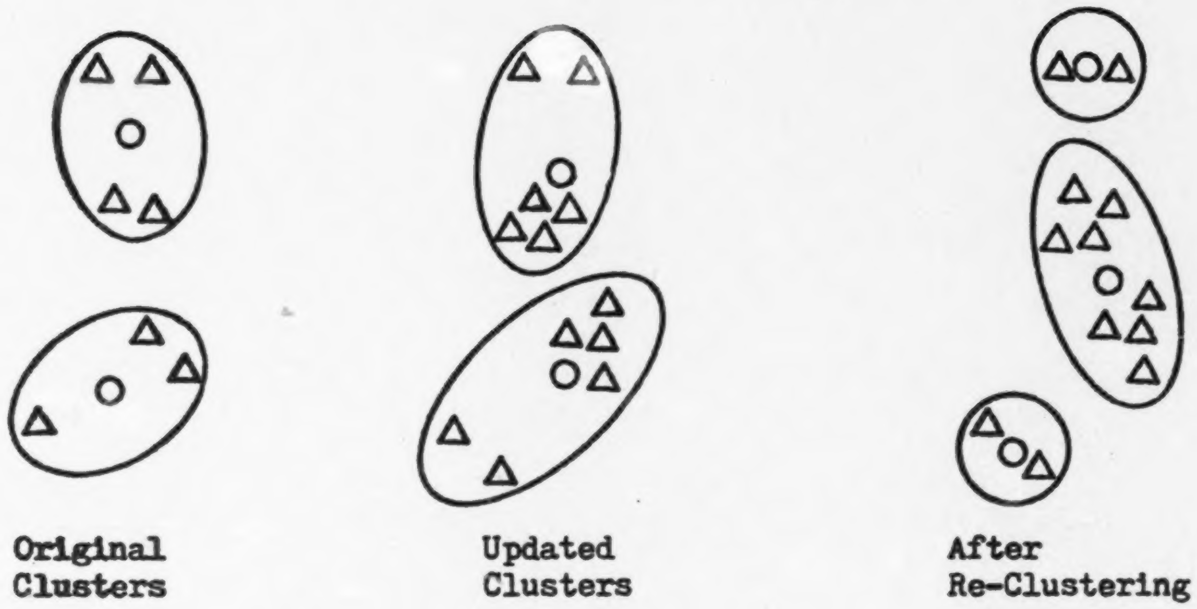
For the most part, SMART's cluster search scheme is appropriate for this investigation, since the experiments focus on profile definition and behavior rather than search strategies. SMART provides many options for controlling the scope (broad or narrow), strategy (forward or backtrack), and matching procedure; additional details are given in Chapter IV and in a paper by Williamson, et al (10).

5. Updating

Additions to a clustered file are made by blending a new document into the cluster with which it has the highest similarity. The update process uses a new document as a request and makes a very narrow search of the file, recording each node that is expanded (update path). The new item is stored in a data or overflow area (see Chapter II) and logically linked to the profile hierarchy. If structured keys are used, the new item's key is simply one more than the key of the last document already in the cluster; linkage is established by increasing the degree of the parent (see Section III.3.A). This blending update procedure is adequate for a time, but it subjects the hierarchy to the changes illustrated in Figure III-5a. In the case of cluster A, its profile continues to characterize both the original and new information; that is, it remains at the center of the cluster. The updated version of cluster B shows polarization



a) Cluster Updating without Profile Modification



b) Cluster Updating with Profile Alteration

Legend △ document ○ profile

Figure III-5

into distinct regions. Finally, cluster C is "enlarged" by the addition of a document which does not really belong in any existing cluster, but which fits best in C. In the last two cases, the profiles no longer accurately represent their respective clusters and to some degree the entire document classification is no longer a logical division of the collection. These intuitive arguments simply point out that in all probability file updating involves two steps:

1) alterations to each profile on the updating path and
2) final incorporation of the new document into a lowest level cluster.

Even if profiles could be altered in an optimal manner the hierarchy continues to degenerate. Figure III-5b shows an example in which two clusters become polarized with addition of new documents. That is, their profiles move to one end of the cluster to represent the majority of documents, and leave the other items virtually unretrievable. No profile can adequately represent these clusters for searches. Furthermore, the example indicates the original classification is no longer a logical division of the data. In light of the new documents three clusters are preferable to the original two clusters. The cause of these problems is the use of a static hierarchy structure to represent a dynamic document collection. Clearly, new items change the character of the data base, and the hierarchy structure must change as well as the profiles. The solution to these problems requires some type of re-clustering whenever the file enlarges enough to cause a significant drop in retrieval performance measures.

The investigation of the updating process has two aims: 1) to examine profile alteration procedures to distinguish those which are effective and 2) to determine how quickly the hierarchy degenerates. That is, to find how quickly precision and recall decrease with increasing file size and thereby obtain an idea of when the file must be re-clustered. The experiments examine three options for altering profiles. The first completely re-makes the profile by re-weighting and adding new terms, and writing the new (probably longer) vector into another part of storage. This process results in an accurate profile, but requires a large amount of work and fragments whatever organization is used in storing profiles. The second option simply associates the new document with the lowest level node and leaves all profiles unchanged. A third option, useful with weighted profiles, alters only existing profile terms and changes their weights to reflect the addition of new documents. No new terms are added, thereby maintaining the original vector length and the storage sequence, since the new profile exactly over-writes its previous version.

With an increasing number of additions to the file, the quality of the classification decays and it becomes desirable to re-cluster the data. Unfortunately, a complete clustering is an expensive procedure and cannot be undertaken too frequently. A most promising technique is partial re-clustering, that is, re-classifying only those portions of the hierarchy that experience significant growth. Under most circumstances many clusters receive few additions and therefore might survive for considerable time without re-organization. More volatile subject areas need frequent revision. In order to implement partial re-clustering, the profile for each node should include the number of documents in its current crown as well as the number of documents added since the last classification (update count). Whenever the ratio of update count to crown size exceeds a specified threshold, all items beneath the node are re-clustered, the update count it reset to zero, and a new update cycle begins. The experiments conducted in Chapter VII investigate parameters applicable to this procedure.

The addition of new documents to a clustered file also increases search time since a new item is stored in an overflow area away from the rest of its cluster. Consequently expanding an upper level node may involve several disk accesses to fetch all the documents in its crown. The effort can be reduced, however, by re-writing the file periodically so its physical and logical sequence are the same. Although there is some expense involved, this procedure consists of moving data only, not restructuring the file. For this reason, this problem is not considered in detail in later chapters.

Another problem in large data bases is the handling of records which are old and essentially inactive. Nearly all information loses value with age; in document retrieval systems, this means that fewer users ask for or accept older documents even if they are relevant. For this reason it is reasonable to retire older items to archive storage--magnetic tape, for example. One way to accomplish this is to make periodic scans of the hierarchy and to remove all documents beyond a certain age. Naturally profiles must be adjusted to reflect deletions, and partial re-clustering may be advisable in cases of significant alterations. The retired documents are maintained as a serial file for retroactive searching. The undesirable features of this scheme are the need for a complete file scan and the loss of information in making the serial file (i.e. the cluster relationships).

Another solution to the problem of aging files is the generation of time dependent clusters. To explain, assume that a 2 year period is chosen and that all information obtained in that period is grouped into lowest level clusters. Let the rest of the hierarchy be made by grouping cluster profiles in the usual manner. The result is a collection divided by subject and publication period (on the lowest level only). Updating occurs only in clusters of the appropriate period. This idea has several advantages. First, old information is easily identified and removed for storage elsewhere. Second, users limiting queries to recent material are able to eliminate search time spent on unwanted documents. Third, the structure of retired information is retained since entire clusters and their profiles are removed intact. The profiles can still be used as a directory so the archive file merely becomes an extension of the disk. Fourth, vocabulary changes might be implemented across publishing periods by associating a dictionary with each period. For example, suppose a thesaurus is used in the indexing process and when the period ends, the thesaurus is reviewed and changed if necessary. Instead of re-indexing all previous documents using the new thesaurus, the old thesaurus (or a list of changes from new to old) is stored along with all other data for that publishing period. As a result, only retroactive searches covering that period must have their queries indexed in terms of the old vocabulary. New documents and requests for more current information are processed using only the newer thesaurus and the data for the new publication period. Because the test collections are not large enough, it is not possible to accurately investigate questions relating to aging files and vocabulary changes. This discussion is included for completeness and to point out an area in need of further research.

6. Hierarchy Storage

A. General

Search speed is one of the primary goals of any file organization. For a clustered document collection stored on disk and accessed by the indexed sequential method, search speed depends on the management of disk space and the storage scheme for documents and profiles. The characteristics of the disk device (Table I-1 for the IBM 2314) and the amount of system traffic (20) also influence retrieval speed; however, these factors are rarely under control of a system designer and, therefore, are not considered in detail here. In this research, management of disk space refers to the allocation of areas for index, data, and overflow purposes within the indexed sequential access method. Lum, et al (21) consider many of these options including

1) number of indexes and their placement;
2) placement and blocksize of data;
3) amount and placement of overflow (cylinder overflow, overflows located on the same or separate disk pack).

In the experiments related to hierarchy storage, it is assumed that any master and cylinder indexes are core resident as well as the track index for the current cylinder. Consequently, a search which crosses cylinder boundaries requires an extra read to fetch a new track index. Overflow space is limited to 10% of the disk capacity, allocated as 2 tracks per cylinder.

The interesting questions concerning hierarchy storage deal with the method and order of record storage. This chapter introduces the concept of structured keys for identifying and linking nodes and suggests three record storage sequences (subtree, heir-filial, and level). To initially build the file or to re-organize it, records are sorted by key value, blocked, and written onto prime data areas of the disk. If each cluster completely occupies an integral number of tracks, there is no wasted space and each access obtains the maximum amount of useful data. In practice, cluster sizes vary a great deal and it is necessary to trade retrieval speed for wasted space. In each tradeoff, there are two opposing ideas to consider. The neighboring concept holds that minimal I/0 delays occur if records in a filial set occupy the minimum number of disk tracks. Filial items are those accessed and processed together such as documents in the same cluster, sons of the same hierarchy node, and all profiles on level 1 (sons of a dummy node on level 0). The splitting concept holds that the amount of wasted disk space is minimized only by splitting records and filial sets across tracks to insure the use of every byte of storage. Obviously minimizing the search time wastes disk space and vice versa. The following storage algorithm attempts to maintain a time-space tradeoff by using a threshold for wasted disk space. It assumes full track blocking (one physical record per track) and stores items in a specified sequence.

B. A Disk Storage Algorithm

The following disk storage scheme is used in the experiments in this research. As mentioned above, it attempts to maintain a balance between space waste and search time. It requires that the order of record storage be known; that is, which record is first, which is second, etc. As items are stored, the necessary indexes for indexed sequential access are inserted whenever a cylinder boundary is crossed. The algorithm consists of three steps repeated for each filial set of records.

1) Let T = track capacity,
   θ = threshold for wasted track space,
   E = remaining space on the current track, and
   S = total size of all records in the current filial set, F.
2) Compute $X = \frac{S}{T}$ = number of tracks to store F if a new track is begun and
   $X' = \frac{S-E}{T}$ = number of tracks to store F if the present track is continued.
3) Store F taking the actions specified below

	Case A	Case B	Case C
Conditions	$X'=X$	$X'>X$ $E \leqslant \theta$	$X'>X$ $E>\theta$
Actions			
Starting storage location	current track	new track (insert ISAM index if required)	current track
Results			
Wasted space	0	E	0
Accesses to fetch F	X	X	X + 1

The frequency of cases B and C depend heavily on θ; the frequency of case A is influenced by the size of the average filial set. After storing the entire file, the important evaluation criteria are the total amount of wasted space and the number of filial sets requiring an extra access (case C).

C. Orders for Hierarchy Storage

The final storage consideration is the order in which records are placed on disk. Of the three sequences mentioned earlier (subtree, level, and heir-filial) and illustrated in Figure III-1, only two are really viable. Subtree order is easily discarded because it locates filial records in widely separated disk areas. Both the level and heir-filial sequences place filial records in close proximity. Order by level stores all nodes on a complete hierarchy level as a contiguous set of physical records. This scheme is most advantageous for broad searches since many nodes are expanded and their proximity reduces the motion of the access arm. As a bonus, order by level allows a hierarchical arrangement of memory devices. For example, the more active, upper level nodes might be allocated to a drum while the less active, lower level nodes reside on moveable head disk. Heir-filial order provides for rapid narrow searches since the sons of a node are generally nearer their parent than with order by levels. However, I/O time for broad searches is high since unrelated profiles on the same level are separated by a moderate distance. Consequently, if several unrelated nodes on the same level are expanded, the disk arm jockeys back and forth among these locations for the rest of the search.

If search strategies are of the forward type, then choice of the optimal storage sequence appears to hinge upon the frequency of narrow and broad searches. This assumes, however, the existence of realistic strategies which take advantage of any economics offered by either scheme. The experiments in Chapter VII consider the applicability of the level and heir-filial sequences for forward search strategies. The storage algorithm described earlier is used to place records in a simulated file prior to actual request processing. An additional outcome of the tests is an estimate of the I/O activity involved in a cluster search and its relation to precision-recall performance.

7. Query Clustering

To this point, the classification procedure has been applied only to document vectors. However, any type of data can be clustered, and query clustering has been suggested as an alternate way of partitioning a document collection (13, 14). Implementation of query clustering requires saving requests and possibly a record of their relevant documents. Using this data, the query vectors are clustered just as though they were documents; therefore the profiles in the resulting hierarchy are combinations of query terms. Second, documents are associated with the lowest level query clusters using one or more of the following schemes.

a) All documents relevant to one or more queries of a query cluster are associated with that cluster.

b) All documents highly similar to a query cluster profile join the corresponding cluster.

c) All documents highly similar to one or more queries of any query cluster join that cluster.

Documents not meeting any of these criteria are clustered in the standard way. The search process using a query generated hierarchy is no different than with document clusters. The success of query clustering depends on the fact that new requests are more likely to be similar to previous requests than to documents. In view of the large difference between speaking and writing vocabularies, it appears plausible that users requesting similar information may phrase their questions similarly so that the success of one user can be passed onto another.

Further, by accumulating requests over long periods of time and by constantly placing relevant documents in the proper cluster, it may be possible to assimilate vocabulary changes in the system on a gradual basis. Early tests with query clustering have shown the technique to be promising, but the results are too few to be very conclusive.

Finally, comes the idea of combining query and document clusters in the same system. Keen (15) describes a search procedure which tries query clusters first and then uses document clusters if the initial search fails or leaves the user unsatisfied. However, there is no real reason to maintain two distinct hierarchies. The nodes of both trees could be combined and searched at the same time. The combined system might be more effective than either one used separately.

8. Alternate Uses of the Hierarchy

The expense of various classification procedures is indicated in

Section III.2. Whatever its cost, clustering is justified only if

1) the time saved in searches over the hierarchy lifetime compensates for the clustering expense or
2) special services or advantages are apparent.

In the second instance, costs are spread among several applications or are justified by increased user satisfaction. This section discusses supplementary uses of a cluster hierarchy and introduces a set of experiments on automatic query alteration. It is not the case that these applications could not be developed using different methods, but having a clustered collection makes them more effective or more economical than might be possible otherwise.

A. Suggested Uses

Because of the classification process, the hierarchy provides a partition of the collection which is far from random. The aim, in fact, is to bias clusters in favor of retrieving relevant information. To some extent, this requires grouping documents with similar subject content. Most of the alternate hierarchy uses--browsing, query expansion and alteration, SDI, and content analysis--benefit from this type of division also.

Viewed from its top, the hierarchy first separates literature into general and then more specific areas on successive levels. This is a natural structure for allowing on-line users to browse among nodes, seeking new areas of interest and discovering new relationships among familiar topics. Depending on the mode of operation, profile terms might be displayed along with substitutes, related terms, indicators of importance, pointers to related nodes, etc. Such information might be used to directly access documents or to refine the keywords used in the current query formulation.

A number of systems employ retrieval thesauri to broaden requests by replacing or supplementing keywords with class descriptors, thereby providing more opportunities for keyword matches. Current research is aimed at automatic thesaurus construction in lieu of the manual methods used previously (16, 17). In many respects, thesaurus construction is simply the process of finding highly co-occurring terms and making them into a single entity. Since clustered documents share a common vocabulary, the cluster is a natural starting place for identifying initial term classes (18). In this way the most obvious and important local associations are recognized first and tested later for their global applicability. Some of the problems caused by high frequency terms are lessened by this procedure.

Syntactic and linguistic analyses also benefit from the fact that a cluster contains homogeneous subject matter. In a sense, the information within a cluster provides a context for interpreting meaning, assigning parts of speech, resolving ambiguities, etc. Since current techniques seem to work best in narrow subject areas, again it is appropriate to apply them after documents are placed in individual clusters.

Procedures for selective dissemination of information (SDI) might also benefit from a clustered collection. Generally these systems store user interest profiles and compare them with all incoming documents. When a profile matches a document, notification is sent to the user so that he is made aware of the document's existence. With a clustered file, a user's name is associated with one or more nodes of the hierarchy. He receives a notification each time one of his nodes appears on the update path of new documents. An advantage of this procedure is that new items are not compared with all user profiles. In addition, names can be associated with upper and lower level nodes, thereby indicating general or specific interests. Finally, only a slight amount of extra work is required to provide SDI since all the comparisons must be made anyway in order to update the data base. To identify which nodes are associated with a user, his original interest profile is used as a query and a file search is made to identify nodes which correlate highly with it. After adding his name to lists for the indicated nodes, the profile can be discarded and SDI proceeds automatically.

B. Query Alteration

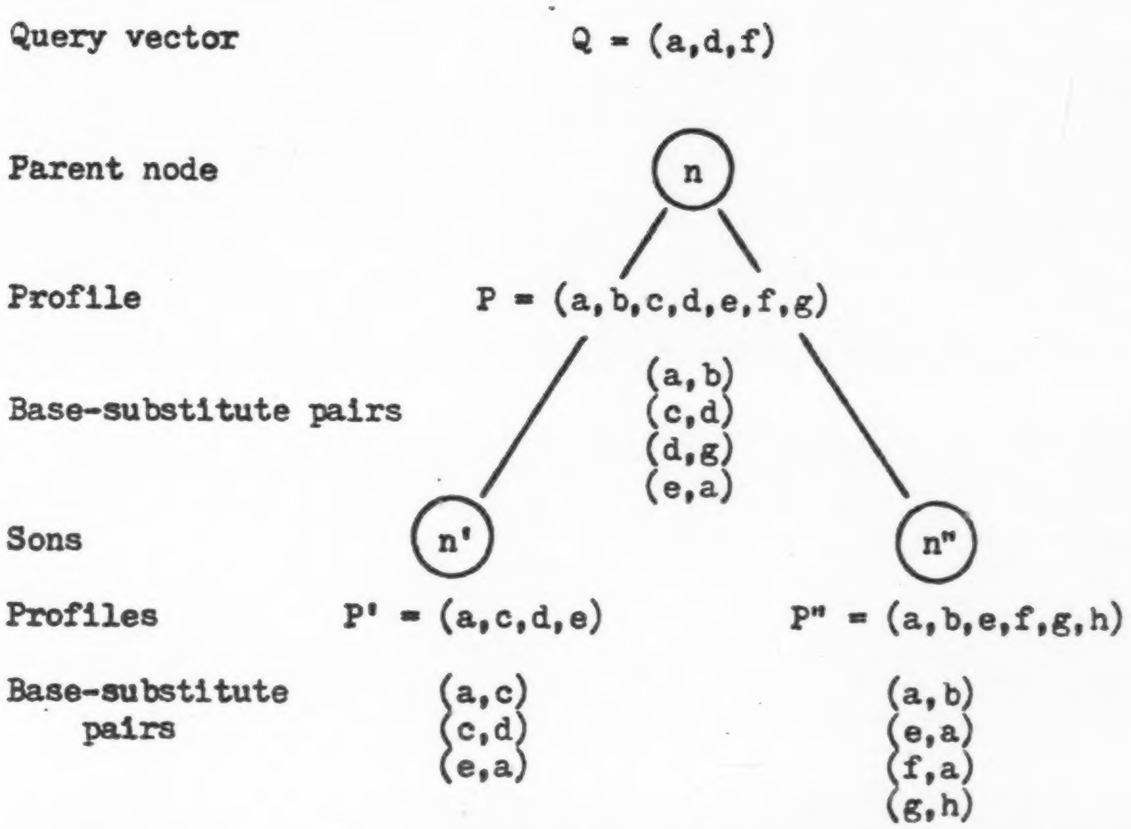
Term classifications have been used by a number of researchers in attempts to improve performance by adding related index terms to documents and requests (16, 17, 19, 22). In some of this work, statistical methods are employed to determine the best substitute for each vocabulary term. Then the base terms in individual vectors are augmented with substitute terms which are used in a wide variety of ways during matching. For example, substitutes might be employed in either requests or documents, or both. Or matches involving substitutes might be counted only if there are matches on their corresponding bases. Or matches might have different emphasis according to whether they involve base or substitute terms. With the proper options, an impressive degree of success has been achieved with these techniques in unclustered collections.

One difficulty in applying these procedures to large collections is the amount of computation for obtaining the base-substitute pairs, for example, preparing a similarity matrix. In this research, a set of substitutes is associated with select profile terms for each node in the hierarchy. Specifically, base-substitute pairs are profile terms which have a maximum term-term correlation in those documents beneath the node under consideration. For the lowest level nodes, only a few terms and documents are involved and similarity matrices are easily calculated and stored. On upper levels, the required matrices are obtained by combining those on lower levels. These techniques greatly decrease the amount of effort in obtaining substitutes, especially when minor profile terms are removed from consideration.

A further bonus from a substitute structure within the hierarchy is its close association with the document collection. The broad substitutes on upper levels are applicable to the entire collection while those on lower levels are concerned with local associations. Each set of substitutes is applied only as query searches enter the appropriate portion of the hierarchy. On any particular level, the substitutes brought to bear may be those from parent nodes (broader terms which expand the search) or those in the current profiles (narrower and more discriminating terms). In either case, the request expansion is temporary in the sense that a new substitute set is applied as the search descends the hierarchy. In addition, there are two general ways of relating base and substitute terms during the vector matching process (TIED and UNTIED options). TIED substitutes have a conditional presence in the expanded vector in that they enter the correlation process only when there is a match on the corresponding base. In a sense, their matches are "tied" to base matches. With the UNITED option, bases and substitutes are treated independently and both are always available for matching, much as in traditional query expansion. Finally, different emphasis may be assigned to matches on base terms and those on substitute terms.

Figure III-6 illustrates these concepts on a small structure--showing the profiles, base-substitute pairs, and the use of substitutes in searching. Not all profiles contain the same terms, so naturally the substitute sets differ for each node. The pair (e,a) is always present, other pairs occur only once or twice. Some terms have the same substitute on both levels while others change substitutes. In the example, the scoring function is simply the number of matching terms. Without the use of substitutes, the request matches profiles P' and P" equally. Using substitutes from the upper level, the query expands to (a,b,d,g,f) and the matching favors node P". Note that in this case, the tied option neglects the match on term g since there is no match on its base d. Two versions of the query are formed when substitutes on the same level are used. Again node P" is favored because of the extra match on term a due to its close association with term f.

The experiments in Chapter VIII restrict themselves to the TIED matching option. The tests are further divided into those for increasing recall by using substitutes from previous levels (parents) and those for increasing precision by using the substitutes for profiles on the current level. Several sets of substitutes are made with varying degrees of frequency restrictions applied to the terms. In all cases the purpose of



Source of Substitutes	Expanded Query	Matching Option	Matching Score	
			P'	P"
None	(a,d,f)	None	2	2
Upper levels Node n	(a,$\underline{b}$,d,$\underline{g}$,f)	Tied	2	3
		Untied	2	4
Current level Node n'	(a,$\underline{c}$,d,f)	Tied	3	
		Untied	3	
	(a,$\underline{b}$,d,f,$\underline{a}$)	Tied		4
		Untied		4

Note: Substitute terms are underlined, base terms are not.

Use of Term Substitutes in Cluster Searching

Figure III-6

the experiments is to show additional uses for the profile hierarchy and to help justify the expense of constructing a clustered file.

9. Summary

This chapter presents a general, but comprehensive review of the construction, use, and maintenance of a clustered file. Clustering methods are classified as generative or divisive and according to the amount of work they involve. Dattola's algorithm is explained in some detail since it is used extensively in the research. The description of hierarchy formation includes cluster size and overlap, linkage among nodes, and profile definition. In particular, the standard profiles and rank value profiles are defined, illustrated, and compared. Next, various search strategies are explained followed by some ideas on how the hierarchy should be stored in order to achieve a fast search. In discussing updating, it becomes apparent that file maintenance involves changes to profiles as well as periodic partial reclustering as the collection grows. The possibility of query clustering is introduced as another way of achieving a viable document classification. The comments on browsing procedures, thesaurus construction, SDI, etc. relate alternate ways of using clusters once they are formed. Of particular interest is automatic query alteration based on term-term associations found in each node's profile.

References

1. G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill, Inc., New York 1968.

2. R. R. Sokal, P. H. A. Sneath, Principles of Numerical Taxonomy, Freeman, 1963.

3. R. Casey, G. Nagy, Recognition of Printed Chinese Characters, IEEE Transactions on Electronic Computers Vol. EC-15, No. 1, February 1966.

4. D.R. Hill, A Vector Clustering Technique, Proceeding of the FID-IFIP Conference on Mechanized Information Storage, Retrieval, and Dissemination, North-Holland Publishing Company, June 1967.

5. D. B. Johnson, J. M. LaFuente, A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering, Report ISR-18 to the National Science Foundation, Department of Computer Science, Cornell University.

6. H. P. Friedmen, J. Rubin, On Some Invariant Criteria for Grouping Data, Journal of the American Statistical Association, December 1967.

7. B. Litofsky, The Utility of Automatic Classification Systems in IS&R, Doctoral Thesis, University of Pennsylvania, 1968.

8. L. B. Doyle, Breaking the Cost Barrier in Automatic Classification, SDC Paper SP-2516, July 1966.

9. R. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

10. D. Williamson, R. Williamson, M. Lesk, The Cornell Implementation of the SMART System, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

11. J. J. Rocchio, Document Retrieval Systems--Optimization and Evaluation, Harvard University Doctoral Thesis, Report ISR-10 to the National Science Foundation, March 1966.

12. G. K. Zipf, Human Behavior and the Principle of Least Effort, Addison-Wesley, Cambridge, Mass., 1949.

13. V. R. Lesser, A Modified Two Level Search Algorithm Using Request Clustering, Report No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, June 1966.

14. S. Worona, Query Clustering in a Large Document Space, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

15. E. M. Keen, Search Matching Functions, Report ISR-13 to the National Science Foundation, Department of Computer Science, Cornell University, January 1968.

16. M. E. Lesk, Word-Word Associations in Document Retrieval Systems, American Documentation, January 1969.

17. K. S. Jones, D. M. Jackson, The Use of Automatically-Obtained Keyword Classifications for Information Retrieval, Information Storage and Retrieval, Vol. 5, pp 175-201, 1970.

18. R. Dattola, D. Murray, An Experiment in Automatic Thesaurus Construction, Report ISR-13 to the National Science Foundation, Cornell University, January 1968.

19. K. S. Jones, E. O. Barber, What Makes an Automatic Classification Effective?, Technical Report, University Mathematical Laboratory, Cambridge, England, 1970.

20. J. Abate, H. Dubner, S. Weinberg, Quereing Analysis of the IBM 2314 Disk Storage Facility, JACM, Vol. 15, No. 4, October 1968.

21. V. Lum, H. Ling, M. Senko, Analysis of a Complex Data Management Access Method by Simulation Modeling, Information Services Department, IBM Research Laboratories, San Jose, California, 1970.

22. K. Sparck Jones, Automatic Keyword Classification for Information Retrieval, Butterworths, 1971.

Chapter IV

The Experimental Environment

1. Introduction to the SMART System.

The principal tool used in this research is the SMART information retrieval system at Cornell University (1, 4). By virtue of its modular design and extensive facilities for gathering evaluation statistics, SMART is more than a simple document retrieval or text processing system. In reality, it provides a laboratory environment for testing the effectiveness of content analysis methods, search strategies, file organizations, and on-line procedures. The system makes available several static document collections with corresponding query sets and a wide variety of processing methods and controlling parameters. Experiments are generally carried out by changing a single variable and making pairwise compararisons between retrieval runs. Some types of experiments suffer because there is no actual user population. However, the advantages of reproducible results and a completely controlled environment more than compensate for this in many cases.

A great deal of the SMART evaluation is based on retrieving user-specified relevant documents. For the most part, the author of each request has carefully examined the collection and identified items which answer his question. These relevance judgments are recorded and used as a standard for measuring the quality of output from experimental tests. Typically, the system correlates a request vector with all or part of a document collection and ranks items in order of decreasing similarity; that is, in the order they are to be retrieved. Using the relevance judgments, the output is scored according to how many relevant are found, at what rank positions, etc. Averaging performance measures across an entire query set provides at least one basis for comparing retrieval methods. There are a number of problems with this evaluation, but its strongest point is that the measures of retrieval quality are based on documents that requestors previously designated relevant and not some internal criterion.

In the usual case, SMART automatically extracts the information content of a natural language document or query and represents it as a vector of weighted concepts. A concept is simply a numerical identifier for a word, word stem, or phrase that actually occurs in the text. The weight reflects the semantic importance of the concept and generally increases in proportion to the frequency of occurrence. Figure IV-1a is a schematic of a vector of this type. The stored representation of a document is a condensed version of this vector containing only concepts with non-zero weights and augmented by header information and a small piece of retrieval data (Figure IV-1b).

Queries, documents, and profiles all have similar internal formats. Any of these items could be considered a vector is an m-space, having one dimension for each vocabulary term. Given two such vectors--a document and a query, a number of functions might be used to measure their proximity in space and hence the desirability of retrieving the document. Experiments with SMART show that the cosine of the vector angle is a rather good measure for detecting relevance, and this function is used extensively in this research (See section II.2.C and Figure IV-1c).

$D = (w_1, w_2, \ldots, w_i, \ldots, w_m)$

Weight assigned to concept C_i

$= (12, 0, \ldots, 24, \ldots, 12)$

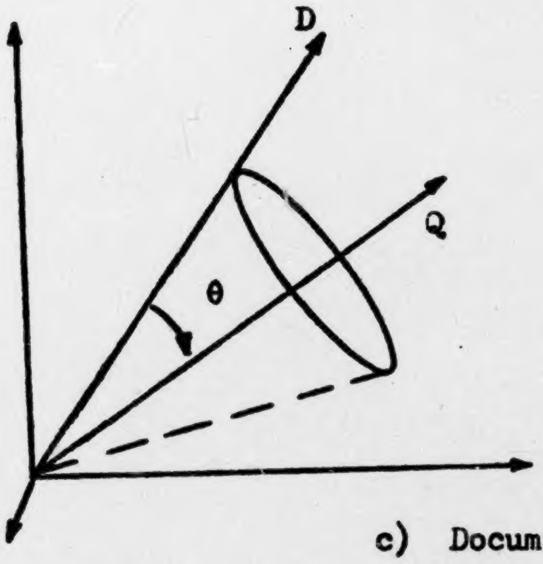
Non-occurrence of concept C_2

a) Document/Query vector

Head	Retrieval Data	c/w	c/w	...	c/w

← 300 Characters →

b) Stored Data Structure



Correlation Function

$$\mathrm{COS}(Q, D) = \frac{Q \cdot D}{|Q| \; |D|}$$

$$= \frac{q_i d_i}{\left[\sum q_i^2\right]^{\frac{1}{2}} \left[\sum d_i^2\right]^{\frac{1}{2}}}$$

Retrieve D if $\mathrm{COS}(Q, D) \leqslant K$ = cutoff

c) Document Space Model

Figure IV-1

With the cosine similarity function, the retrieved items lie within an m-dimensional cone swept about the query vector. The half-angle of the cone is $\cos^{-1}(K)$, where K is the cutoff established to distinguish retrieved and non-retrieved documents.

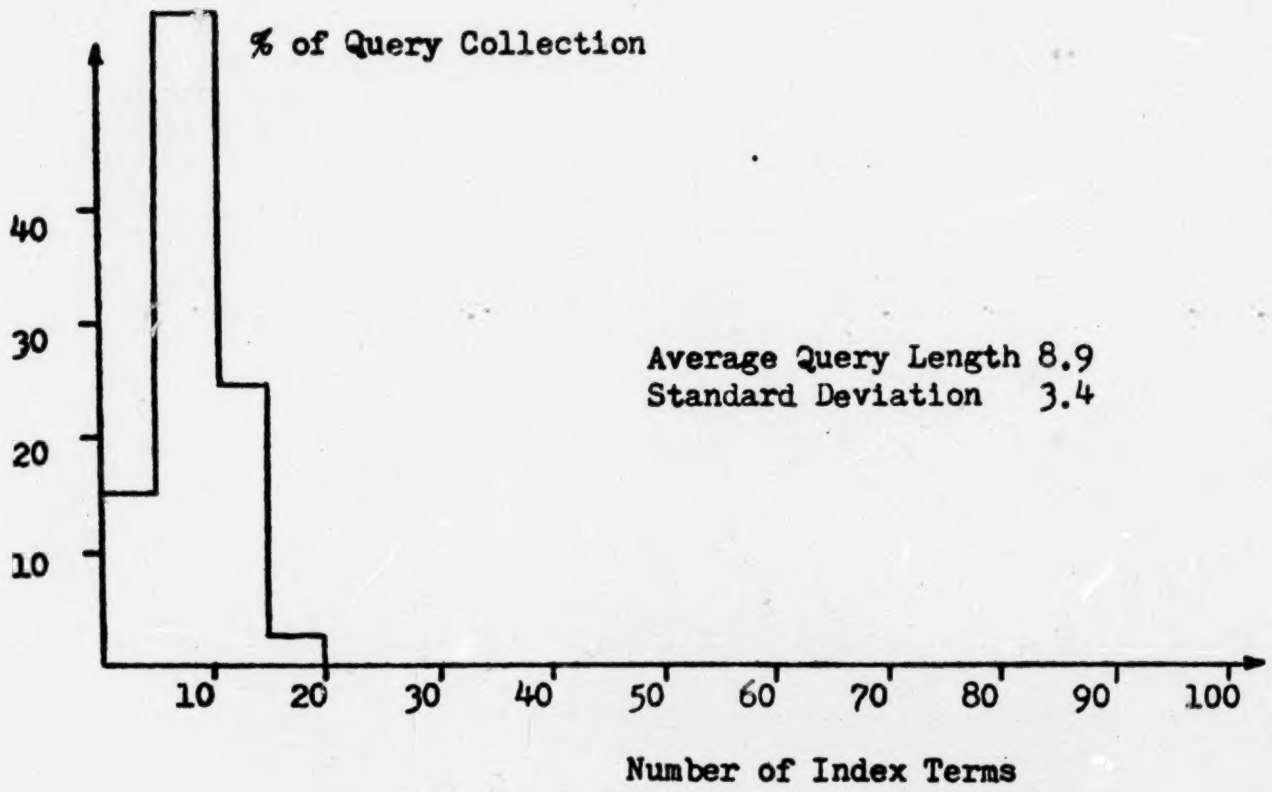
2. The Data Collections

The collection used in this study consists of the abstracts for 1400 aerodynamics articles and their corresponding 225 requests with relevance judgments. Texts for both documents and queries are those used by Cleverdon's Aslib Cranfield Project (2). The SMART word stem analysis procedure was applied to the text in order to obtain search vectors. Specifically, each item is indexed by

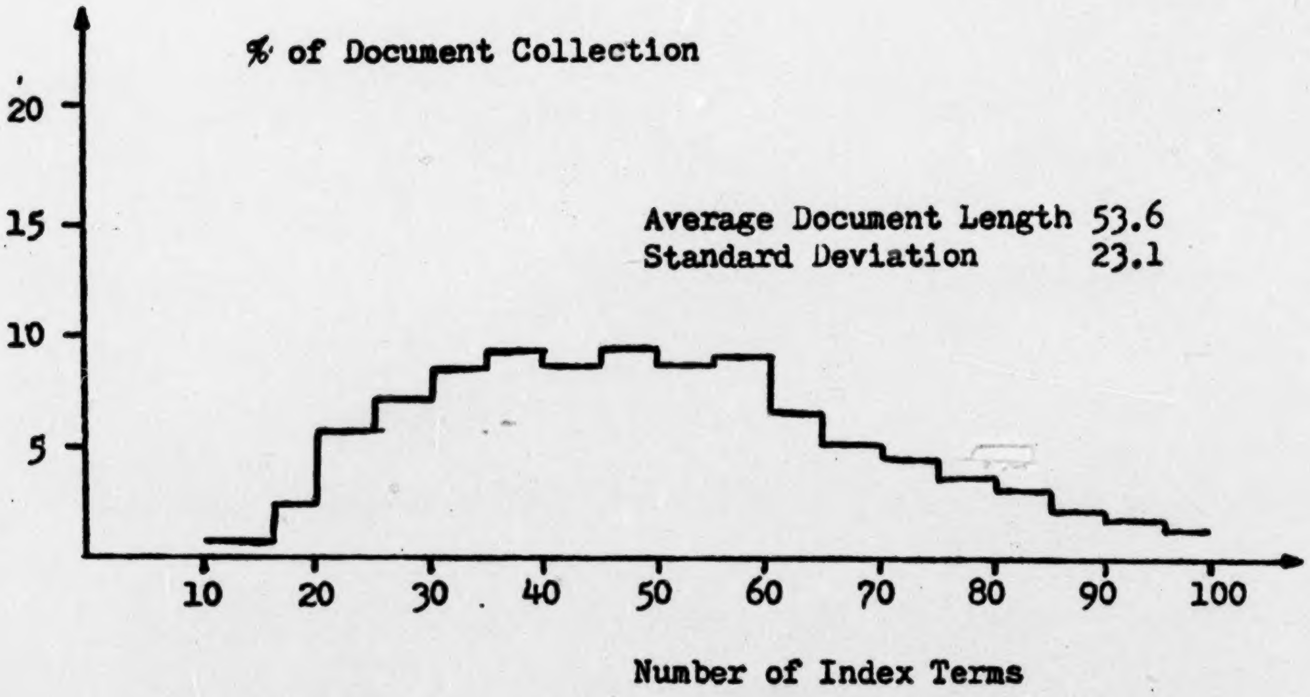
1) deleting words found on a restriction list,
2) reducing morphological forms of the same word to their common stem, and
3) weighting each stem according to its number of occurrences.

The restriction list consists of approximately 360 prepositions, pronouns, conjunctions, and auxiliary and common verbs (See Appendix A). The automatic stemming procedure confounds words ending with common suffixes while taking into account doubled final consonants, changes of y to i, and the removal of silent e's. This processing results in a vocabulary of 5000 words with a rank-frequency distribution similar to that of the well-known Zipf curve.

For later reference, the distribution of document and query lengths is shown in Figure IV-2. In this context the length of a vector refers to



a) Distribution of Lengths of the 225 Cranfield Queries

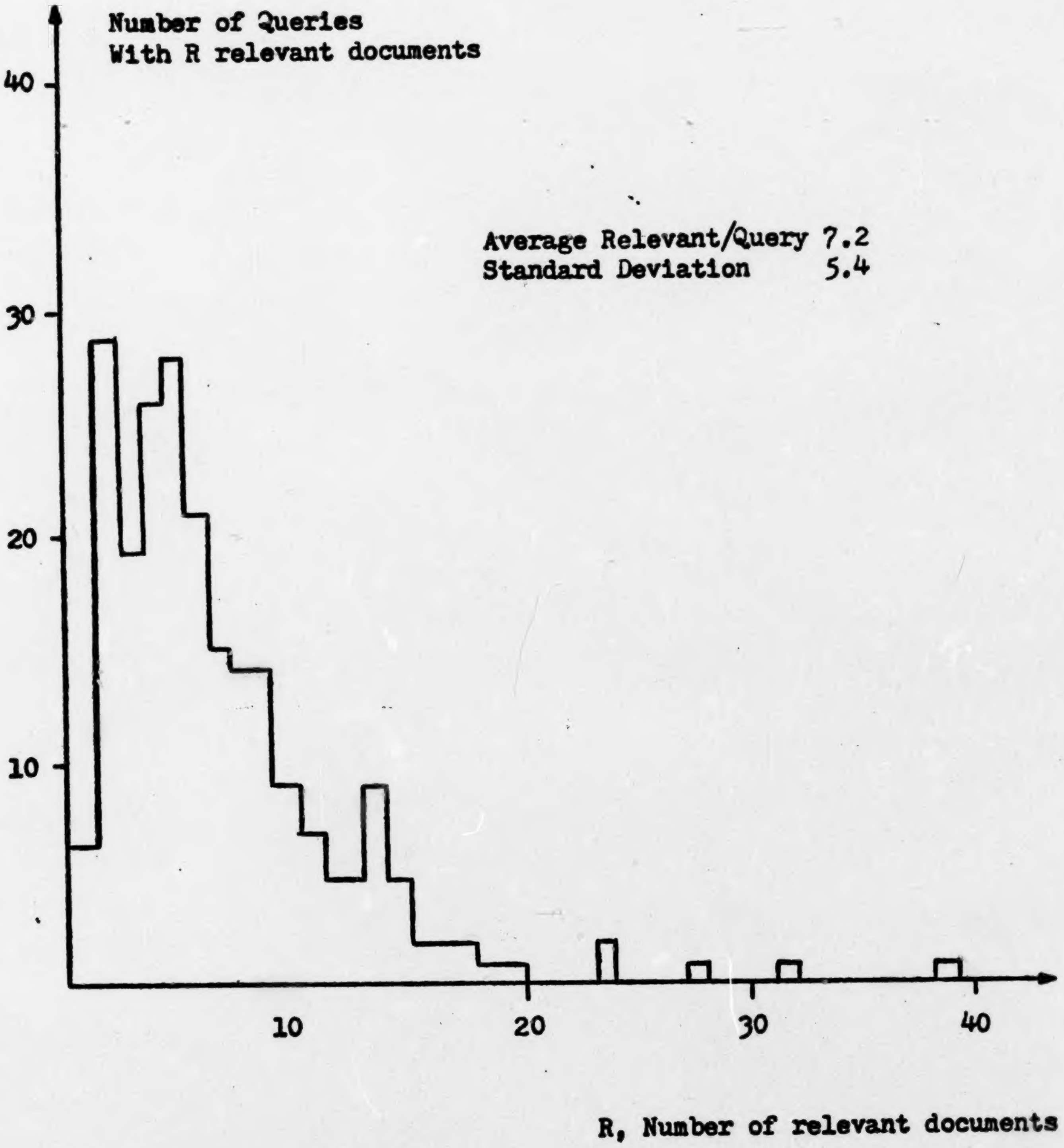


b) Distribution of Lengths of the 1400 Cranfield Documents

Figure IV-2

its number of non-zero elements; that is, the number of index terms assigned to the corresponding text. Understandably, documents are longer than queries, averaging 54 terms as opposed to 9 for queries. A number of documents have in excess of 100 terms. The distribution of weights is just as important as length. Within documents, term frequencies range from 1 to 27 occurrences. A great many terms occur once, a smaller number twice, and so on; the distribution appears to be almost Poisson. Queries differ markedly in that 97% of all terms have unit frequencies within their vectors and the other 3% occur only twice. As it is, query terms might as well not be weighted according to frequency at all. These results indicate that users write short specific requests and omit background material that might really be helpful. This may be a genuine user preference or because instructions are not given to the contrary.

A final statistic to report is the distribution of relevant documents for each query (Figure IV-3). With an average of 7 relevant items per question, the collection generally is 1.2. Using all the averages mentioned so far, the typical query contains 9 terms of the same weight and aims at retrieving 7 relevant documents indexed with 54 concepts apiece. Obviously the task is a hard one, for even if all query terms are matched a document might have a cosine correlation of only 0.41. Under these conditions, random keyword matches are expected to have a noticeable influence on performance. Longer request formulations would help this situation, but unfortunately users do not seem to supply them on their own accord.



Distribution of relevant documents

Cranfield Collection, 1400 Documents, 225 Requests

Figure IV-3

3. The Generated Hierarchies

Three cluster hierarchies were produced for the Cranfield documents using the Dattola classification procedure (3). All consist of three levels--two for profiles and one for documents--but their clusters differ greatly in size and overlap. Hierarchy 1 is the primary test case for this study; the others are used to confirm selected test results. Table IV-1 lists the properties of each hierarchy; in order to interpret this data, a few definitions will be reviewed. The crown of a node is the number of documents reachable from it along all descendant paths. For a particular query, a node is relevant if and only if at least one relevant document is included in its crown. The values reported in the table are the average number of relevant nodes per level. Finally, overlap is defined as the ratio of the total number of leaves to the collection size (minus 1.0 and expressed as a percent).

The first two hierarchies are designed so that each document cluster fits onto a single disk track (approximately); the third allows about two clusters per track. Comparing Hierarchies 1 and 3 in Table IV-1, both have about the same overlap and node degrees, although the latter contains nearly twice as many nodes. Hierarchy 2 has much more overlap, but only a few nodes on level 1 (with high degrees). As a result, its first level profiles characterize 450 documents (indirectly) rather than 50 or 100 as in the other cases.

The number of relevant nodes per level provides a superficial evaluation of Dattola's classification procedure without regard to searching. The table shows that level 2 clusters confine relevant documents to a

Property	**Hierarchy**		
	1	**2**	**3**
Level 1 (Profiles)			
Number of nodes	13	6	28
Average crown	115	446	50
Range of crowns	60-201	267-445	22-81
Average relevant nodes	3.9	3.3	5.2
Average sons	4	16	4
Average P_3 profile length	812	908	526
Level 2 (Profiles)			
Number of nodes	55	94	103
Average crown	27	28	14
Range of crowns	10-55	11-64	4-25
Average relevant nodes*	5.3	9.0	5.1
Average P_3 profile length	323	311	197
Average P_2 profile length	266		
Level 3 (Documents)			
Total number of nodes	1500	2679	1400
Overlap	7%	91%	0%
Average relevant nodes (unique)	7	7	7
Average document length	54	54	54
Range document length	10-164	10-164	10-164

***Based on partial data**

Crown = number of documents reachable from a node
Relevant node = a node whose crown contains one or more relevant documents
Overlap = ratio of total nodes on level 3 to collection size (1400) less 1.0

Properties of the Experimental Hierarchies

Table IV-1

small number of groups, but the level 1 clusters do not confine them a great deal more. It is unfortunate that the algorithm is not able to place all the relevant for a query under a single first level node, especially since a typical query has only 7 relevant documents. In the present situation, the broader search strategies should expand 3-5 nodes on level 1 and 5-9 nodes on level 2. In some circumstances this destroys the economy of involving the first level profiles at all. To illustrate, suppose that all level 1 profiles occupy two disk tracks while all level 2 profiles occupy 5 tracks. If 3 nodes are expanded on each level, then the search cost is probably 2 accesses for level 1 and 2 to 4 accesses for level 2, making a total of 4 to 6 accesses. Under these circumstances, it might be better to disregard level 1, just examine level 2, and incur a fixed cost of 5 accesses. Obviously the situation is improved in both cost and performance if the relevant are grouped more tightly. However in the experimental environment, collection size constrains the number of nodes if a reasonable cluster size is maintained. In an actual collection of thousands of documents, there would be a great many more top level nodes and the economy becomes more apparent.

Table IV-1 also shows the average vector length (number of index terms) for some of the standard profile types in each hierarchy. As large as the vectors are they do not strictly conform to the definitions of standard profiles (see Section III.3) since the definitions generate vectors with more index terms than the software can handle. In order to perform any experiments, all terms with frequency 1 are removed in each P_3 or P_2 vector. Even so, on level 1 of Hierarchy 2 terms of frequency 3 or less had to be eliminated. At this point it is impossible to substantiate that this deletion has negligible effect on results; later analysis and experiments confirm this assertion. Since P_3 profiles are based on term occurrence frequencies and P_2 profiles use document frequencies, the initial term deletion results in vectors of different lengths. Consequently, two sets of unweighted (P_1) vectors are possible, depending on whether P_2 or P_3 vectors are used as starting profiles. For example, given P_3 vectors, their unweighted counterparts are generated simply by setting all weights to a constant value. In an actual system, this storage space would be reclaimed. In any case these shortened, but otherwise undisturbed vectors are used as "standard" profiles throughout the study. The set of unweighted vectors will always be identified and consistent with other profiles used for comparison.

4. Evaluation

This research is a study of profile definitions, uses, and modifications, as well as an evaluation of clustered files in general. However, evaluation is complicated by the following three factors:

1) changes in profile definitions may affect several evaluation measures (storage, search time, quality of search output),
2) output quality varies with the search strategy, and
3) imperfect evaluation measures.

The first factor points out the difficulty of selecting a "good" profile since it is improbable that a single definition maximizes all the desired measures. Output quality refers to the amount of relevant material retrieved and the order of its presentation. These quantities depend on the number of profiles expanded on each hierarchy level (search strategy) which has a secondary effect on response time. Consequently, a "good" profile for one search strategy may not perform well for another strategy. Hopefully, some of these difficulties are eased by the dual evaluation procedure adopted here. Both methods are based on external relevance judgments for the query set. In the first case, the standard SMART precision-recall computations are made for fixed search strategies. The second, new method is based on the content of clusters chosen for expansion rather than the final document order. Although the search is not completed, the method is independent of search strategy. Finally in nearly all cases, comparisons are made among runs with only one changed parameter. With these precautions, the general consistency of both evaluation methods, and the relatively large query set; reasonable confidence is placed in the conclusions drawn from the output.

A. SMART Evaluation

A SMART cluster search strategy includes selections for parameters concerned with

1) measuring query-profile similarities and
2) deciding which nodes are expanded (1).

The cosine correlation is used exclusively in this work although it has been suggested that query-profile similarities should be influenced also by the position of a node in the hierarchy. Some of the results in Chapter V deal with this question. In order to decide which nodes should be expanded, SMART maintains a list of nodes examined on all previous levels and arranges it in order of decreasing similarity. At each expansion point, the list is scanned from its beginning and nodes are expanded until one of the following criteria is met:

1) the cumulative crown of the expanded nodes exceeds a preset maximum,
2) the number of expanded nodes falls within a chosen range,
3) correlations fall beneath a threshold, or
4) the end of the list is reached.

Several additional factors may influence the criteria also. In any case the sons of the nodes expanded are fetched, matched with the query, and their correlations sorted back into the list to await further processing.

In the test situation here, two fixed search strategies are used for evaluation. The first is a narrow search designed to examine 5% of the documents and expand approximately 1 node per level. The second is a broader search looking at 10% of the collection and expanding 1 to 2 nodes per level. The complete set of SMART parameters is given in Table IV-2.

Given a search strategy, SMART examines the most promising profiles and produces a list of documents ranked in order of decreasing similarity. Suppose that the first k documents on the list are retrieved. Then for each query, the document collection is divided into four exclusive sets (Figure IV-4a):

1) retrieved and relevant (a documents)
2) retrieved and non-relevant (b documents)
3) non-retrieved and relevant (c documents)
4) non-retrieved and non-relevant (d documents)

Narrow Search*

Maximum cumulative crown (WANTED)	70
Minimum nodes expanded (MINNOD)	1
Maximum nodes expanded (MAXNOD)	1
Correlation margin about expansion cutoff (EPS)	.005
Minimum correlation threshold (MINCOR)	.05

Broad Search*

Maximum cumulative crown (WANTED)	140
Minimum nodes expanded (MINNOD)	1
Maximum nodes expanded (MAXNOD)	3
Correlation margin about expansion cutoff (EPS)	.005
Minimum correlation threshold (MINCOR)	.01

*All other parameters are 0.

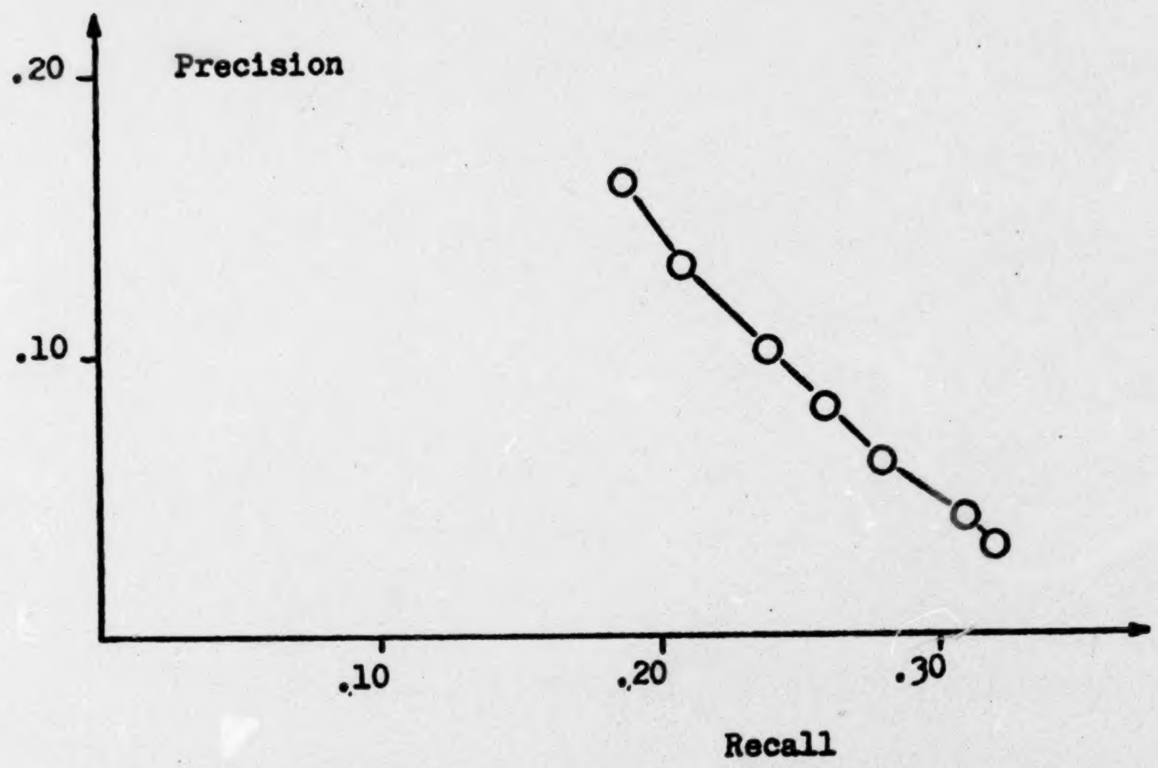
SMART Cluster Search Parameters

Table IV-2

	Relevant Documents	Non-relevant Documents
Retrieved Documents	a	b
Non-retrieved Documents	c	d

k = a+b = number of documents retrieved

a) Subdivision of a Document Collection After Retrieval of k Items



Retrieval cutoffs are 5,10,15,20,30,50,75 documents

b) Sample Precision-Recall Plot for a Cluster Search

Figure IV-4

Note that the cutoff for retrieval is $k = a+b$. Under these conditions the search precision and recall are defined as:

$$P = \frac{a}{a+b} \qquad \text{(IV-1)}$$

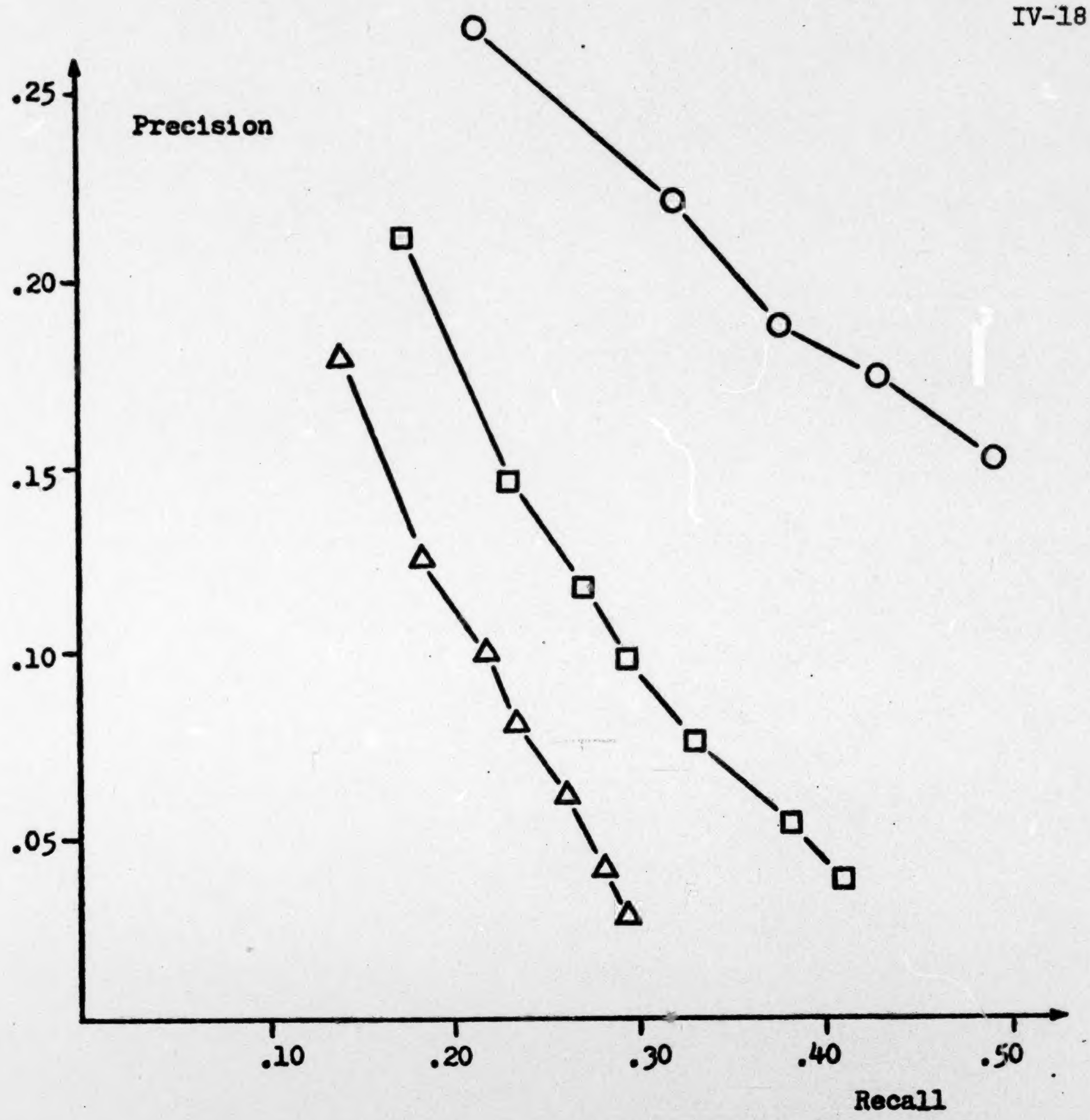
$$R = \frac{a}{a+c} \qquad \text{(IV-2)}$$

SMART evaluation consists of plotting averaged precision-recall data for a range of cutoffs as well as computing four global measures. For this research, P-R points (precision-recall) are calculated for cutoffs of $k = 5, 10, 15, 20, 30, 50, 75$ (the broad search includes $k = 100$). All possible cutoffs are specifically not used in order to avoid placing undue emphasis on the initial P-R values. From a user's view it makes little difference whether a relevant document is ranked first or third since a half dozen items are probably judged anyway. However, differences in early rank positions have considerable influence on P-R curves. By plotting points at spaced intervals, this bias is lessened. A sample curve is shown in Figure III-4b. From equations IV-1 and IV-2 it is seen that recall-precision values of 1.0 are perfect so that the better the performance, the higher the curve.

The global measures calculated are normalized precision, normalized recall, rank recall, and log precision (5). The two normalized values approximate average standard precision and average standard recall, but actually measure the difference between the ideal performance (precision and recall of 1.0) and actual performance. Rank recall and log precision are somewhat simplified forms of the normalized measures. All of these values are strongly influenced by initial data points (ranks 1, 2, 3,...)

and are not easily corrected for this factor. However they are useful for condensing the graphical output into a few statistics for easier evaluation.

Figure IV-5 contains the precision-recall curve and global measures for a cluster search using Hierarchy 1, P_3 profiles, and both search strategies described earlier. Corresponding points from a P-R curve for a full search are included as a basis of comparison for this and future searches. Considering only the relative positions of the curves and neglecting the amount of system effort (user cost) involved, it is easy to conclude cluster searching is vastly inferior to other methods. However, cluster searches are not intended to have the complete effectiveness of a full search. Their usefulness comes from flexibility; in an inexpensive, narrow search a few relevant items can be obtained, and as the search broadens greater recall is achieved. A very broad search might completely neglect the upper levels of a hierarchy and, in the limit, become a full search. Search cost has been treated casually thus far whereas it is an important factor in comparing the results of cluster search strategies. As mentioned earlier, the number of disk accesses per query search is regarded as a reasonable cost measure, at least being proportional to both computing charges and on-line response time. Because SMART handles many queries in parallel and actually simulates cluster searches, it is impossible to obtain the number of disk accesses per query. The average number of correlations per level is known however, although this data is difficult to relate to accesses without knowing the details of record storage. Even if an accurate measure of system effort were available some



Symbol	Description	NR	NP	C(1)	C(2)	C(3)
○	Full Search	.88	.61	0	0	1400
△	Narrow Cluster Search	.62	.35	13	8	90
□	Broad Cluster Search	.67	.42	13	12	160

Legend: NR - normalized recall
NP - normalized precision
C(X) - number of correlations on level X

Sample Precision-Recall Curve for Hierarchy 1

Figure IV-5

subjective judgments are required to settle tradeoffs between cost and performance. All this simply proves that search evaluation is not a straight forward task. Fortunately, the tests ahead compare similar runs with approximately the same number of correlations per level. This greatly simplifies the job of rating various profile definitions, etc.

The SMART evaluation procedure has a number of drawbacks of a mechanical and aestetic nature. First, averaging performance over a query set must provide for different numbers of relevant for each query; interpolation etc. (6). Second, because a cluster search does not examine all documents, it may not have the opportunity to retrieve all relevant information. This condition results in an artifical upper bound on recall or a recall ceiling. In order to reduce the influence of this factor, P-R curves are not displayed beyond the retrieval cutoff. The global measures require ranks for all relevant items, however, and for this reason curves are extrapolated by assuming that unretrieved relevant documents would occupy random positions in the remaining output (7). Third, precision and recall measure user satisfaction without regard to search cost. Attempts have been made to incorporate cost by changing the extrapolation procedure, but for the most part, system effort is recorded by the number of correlations per hierarchy level and these figures are simply associated with a P-R curve. Fourth, the evaluation process is quite dependent on search strategy. As a result, conclusions stated under one set of conditions may or may not apply to a different set of parameters. Finally, the cost of SMART search and evaluation procedures preclude examining a large number of search strategies.

In spite of these drawbacks, evaluation by means of document level precision-recall curves is useful in that it gives complete information about a specific type of search and that curves of this type have become the standard measures in document retrieval systems.

B. Cluster Oriented Evaluation

This section develops a method for evaluating profiles based on their success of differentiating relevant and non-relevant nodes. The measures used--recall ceiling and precision floor--are analogous to document recall and precision, but extended to clusters. They account for overlap and place different values on clusters due to their size or the amount of relevant information they contain. From one point of view, the evaluation considers the retrieval of clusters of information rather than single documents. Accordingly, statistics are computed only after each cluster is "retrieved".

Consider a hierarchy level with m nodes (Figure I-2). The broadest possible search strategy, examining all m nodes, involves all the distinctions to be made among these nodes under any condition. For query i, assume the nodes are ranked by decreasing correlation so that they would be expanded in this sequence. For the node ranked $j = 1, 2, \ldots, m$, let c_{ij} be the number of documents in its crown that are not present in the crowns of nodes with higher ranks. Similarly let r_{ij} be the number of relevant documents in its crown that have not been recovered previously. Note that c_{ij} and r_{ij} compensate for overlap as it is encountered in the sequence of ranked nodes. The sums $\sum_{j=1}^{k} c_{ij}$ and $\sum_{j=1}^{k} r_{ij}$ represent the cumulative crown and cumulative number of relevant over k nodes. The quantity

$$u_i = \sum_{j=1}^{m} r_{ij} \qquad \text{(IV-3)}$$

is the total number of documents relevant to the query. Recall ceiling is the percent of all relevant that are recoverable subject to the expansion cutoff j:

$$RC_{ij} = \frac{r_{i1} + r_{i2} + \ldots + r_{ij}}{u_i} \qquad \text{(IV-4)}$$

This is the highest possible document recall that is attainable for this search strategy. Obviously large values of RC_{ij} are preferred and $RC_{i1} = 1.0$ is the optimal situation--all relevant documents beneath the first node. Deceptively high recall ceilings could be obtained by placing nearly all documents in a single large cluster and dividing the remainder into several small clusters. For most requests, the large cluster is expanded first and a high average recall ceiling is obtained. Viewing only this measure, it appears that good performance is obtained by examining one cluster. This is only partially true, of course. The clusters used here do not have skewed size distributions, but slight effects of size are observable. Precision floor corrects for size bias by measuring the percent of recoverable documents that are relevant, subject to the expansion cutoff j:

$$PF_{ij} = \frac{r_{i1} + \ldots + r_{ij}}{c_{i1} + \ldots + c_{ij}} \qquad \text{(IV-5)}$$

Precision floor represents the lowest possible document precision if all documents beneath the first j nodes are retrieved. Again, large values of PF_{ij} are preferred; $PF_{i1} = 1.0$ indicates the ideal situation--all relevant and no non-relevant beneath the first node.

As an example of these measures, consider a hierarchy level having 5 nodes with non-overlapping crowns of sizes 10, 10, 20, 20, and 30 documents respectively. Suppose that a query with 8 relevant documents is correlated with the profiles and produces the ranking and evaluation statistics shown in Figure IV-6. The values are interpreted as follows. If only one node is expanded, regardless of what else occurs in the search, the highest possible document recall is 4/8 and the lowest possible document precision is 4/20 (assuming all are retrieved). Similar statements can be made for other cutoffs. The accompanying graphs show the changes in RC_{ij} and PF_{ij} for varying expansion cutoffs.

After processing n queries, the average recall ceiling and average precision floor are computed as follows:

$$\overline{RC}_j = \frac{1}{n} \sum_{i=1}^{n} RC_{ij} \qquad \text{(IV-6)}$$

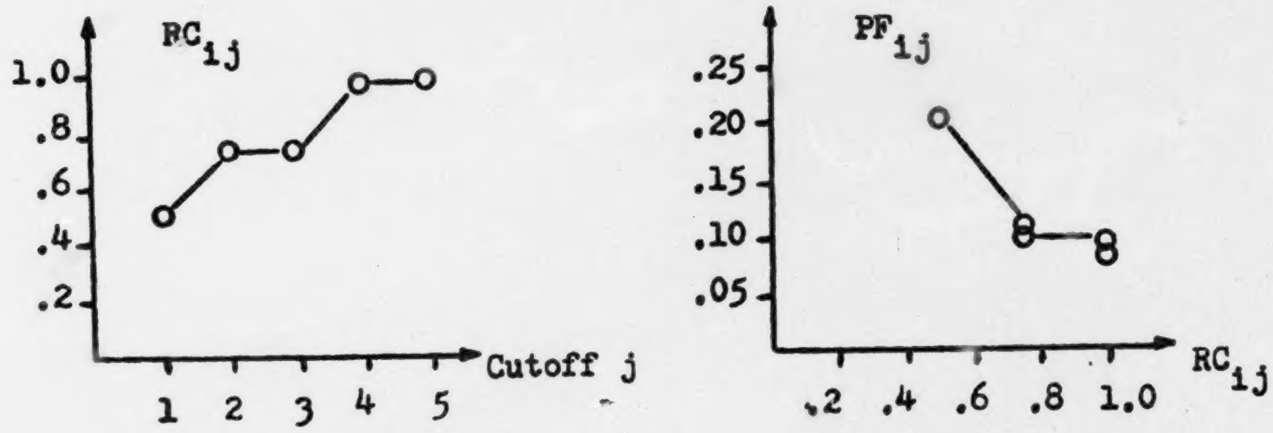
$$j = 1, 2, \ldots, n$$

$$\overline{PF}_j = \frac{1}{n} \sum_{i=1}^{n} PF_{ij} \qquad \text{(IV-7)}$$

These are actually macro averages in that they average the individual performance statistics for each query of the set (6). Other averaging methods could be defined also, but are not used here. Both $\overline{RC}_j$ and $\overline{PF}_j$ can be plotted separately or together along with some measure of system work to provide a performance curve for one level of the hierarchy. Figure IV-7a is a hypothetical plot of $\overline{RC}_j$ versus the number of clusters expanded. All such curves are non-decreasing and achieve a maximum of $\overline{RC}_j = 1.00$ for $j \leq n$. In the ideal situation every query has all its

Rank j	Node	c_{ij}	$\sum c_{ij}$	r_{ij}	$\sum r_{ij}$	RC_{ij}	PF_{ij}
1	4	20	20	4	4	4/8	4/20
2	5	30	50	2	6	6/8	6/50
3	1	10	60	0	6	6/8	6/60
4	3	20	80	2	8	8/8	8/80
5	2	10	90	0	8	8/8	8/90

a) Example of cluster oriented evaluation statistics



b) Graphs of recall ceiling and precision floor for the above query.

Legend

query i - 8 relevant documents

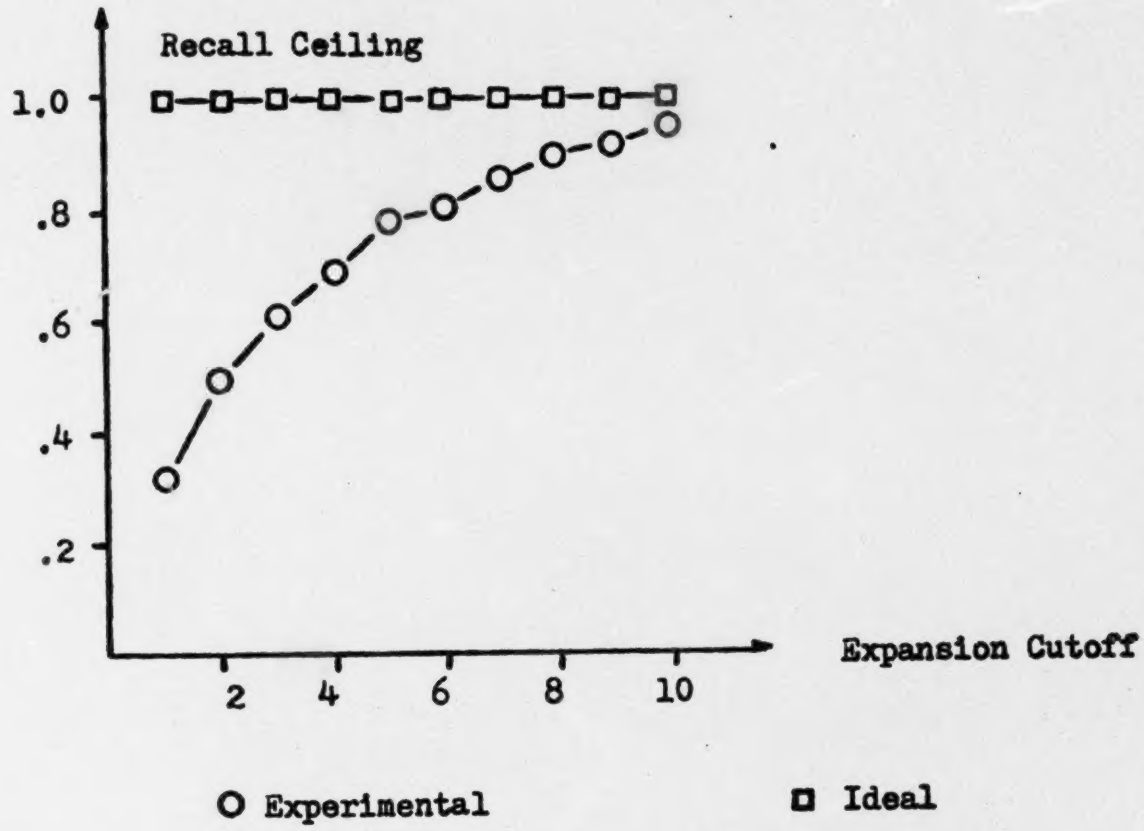
C_{ij} = number of additional documents in the cluster ranked j

r_{ij} = number of additional relevant documents in the cluster ranked j

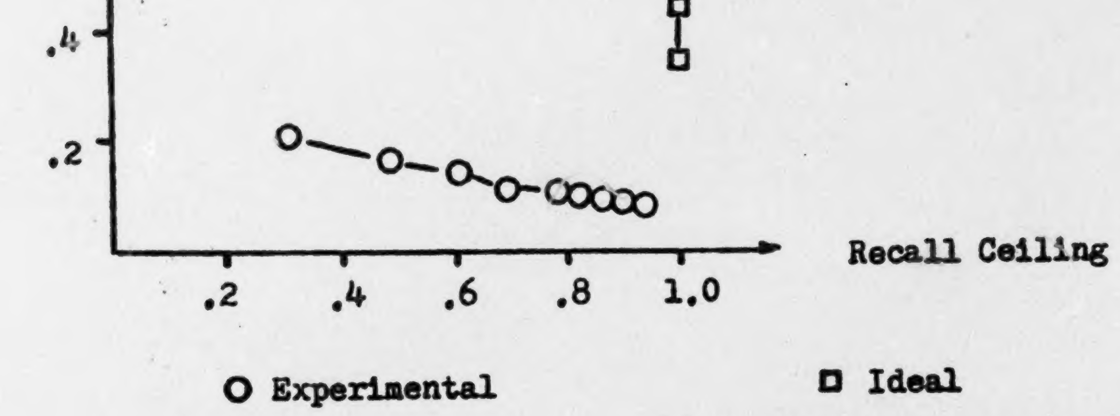
RC_{ij} = recall ceiling if j nodes are expanded

PF_{ij} = precision floor if j nodes are expanded

Figure IV-6



a) Hypothetical Plot of Recall Ceiling Versus Expansion Cutoff



b) Hypothetical Plot of Recall Ceiling Versus Precision Floor
Plotted at various Expansion Cutoffs

Figure IV-7

relevant in the crown of the top ranked node so $\overline{RC}_j = 1.00$ for $j = 1, 2, \ldots, n$. Hence, the aim is to raise experimental curves to a horizontal line. Figure IV-7b is a hypothetical plot of $\overline{PF}_j$ versus $\overline{RC}_j$ drawn at various expansion cutoffs. With this curve, the ideal case arises if for every query, only relevant documents are found in the crown of the top ranked node. Hence $\overline{RC}_1 = \overline{PF}_1 = 1.00$, and the ideal graph is a vertical line at the extreme right of the scale.

Measuring system work (search time and cost) is a non-trivial task, involving the selection of a work unit and problems associated with measurement. The number of disk accesses per search is probably the best unit of work. However measurements made under this condition depend on the characteristics of a specific storage device, order of hierarchy storage, blocking factors, etc. As a result, disk accesses are too specific a unit except where these factors are controlled. The SMART evaluation suggests measuring work by the number of query comparisons made with documents and profiles. The inaccuracies here are twofold. First, size differences among data vectors on various levels indicate that not all comparisons incur the same cost. Second, this unit ignores the economy from storing vectors in adjacent locations. The cluster evaluation scheme measures system effort by the number of nodes expanded in the search. This quantity is device independent and emphasizes the economy from storing items adjacently. A fixed number of accesses--one or two--might be associated with each node expanded in order to provide conversion to other units. However, the varying number of sons per node is neglected and obviously, more work is involved in expanding a node with many sons than with a few sons. In spite of this final difficulty, the number of nodes expanded is used as the measure of system effort in the majority of cases in this study. Basically, it is assumed that averaging over many nodes and many queries minimizes the effects from variations in the degrees of nodes.

Given two profile definitions P_A and P_B, there must be some agreed method of using the recall ceiling and precision floor measures and curves to determine which definition is superior. The following rule is used for this purpose: P_A is said to be superior that P_B if the average values of recall ceiling and precision floor for P_A are greater than the corresponding values for P_B. Symbolically this is expressed as:

$$P_A > P_B \Longleftrightarrow \begin{cases} (\overline{RC}_j)_A > (\overline{RC}_j)_B \\ (\overline{PF}_j)_A > (PF_j)_B \end{cases} \qquad \text{(IV-8)}$$

$$\text{for } j = 1, 2, \ldots, x \leq m.$$

Note that the values of j may be restricted to the initial ranks since $\overline{RC}_m = 1.0$ in all cases.

So far, evaluation considers a single level of the hierarchy and all of its nodes. However, an actual search generally accesses only part of the nodes on any one level. Still, it is reasonable to use all nodes in evaluation since this includes the full set of items to be distinguished under any search strategy. Examining multiple subsets of nodes is possible, but turns the evaluation into an undesirable combinatorial test situation. This is really unnecessary since it can be shown that if $P_A > P_B$ holds for all nodes on a level, then it also holds for a majority of subsets of these nodes. For any particular subset S, either

1) $P_A > P_B$ within S or

2) $P_B > P_A$ within S.

However, since $P_A > P_B$ for the entire level, then the first case must be more prevalent among all possible subsets. As a result, $P_A > P_B$ for the majority of search strategies.

The remaining concern is the interaction among all levels of the hierarchy. The evaluation considers each level independent of the rest. Given two profile definitions P_A and P_B, is it possible to have a contradiction such as $P_A > P_B$ on one level and that $P_B > P_A$ on another level? Although possible, this situation is highly improbable if the profile definition is at all reasonable--assigning term weights which are non-decreasing with the number of keyword occurrences. Further, since profiles on higher tree levels are composites of those on lower levels, it is even more difficult to realize the contradiction if a profile definition is consistently applied. Lastly, if a contradiction occurs it is fairly clear that neither profile has a strong superiority over the other. Both probably perform about the same.

As mentioned either, the ideal case arises when $\overline{RC}_1 = \overline{PF}_1 = 1.0$. This situation occurs if for all queries:

1) the classification isolates all relevant and no non-relevant in a single cluster and

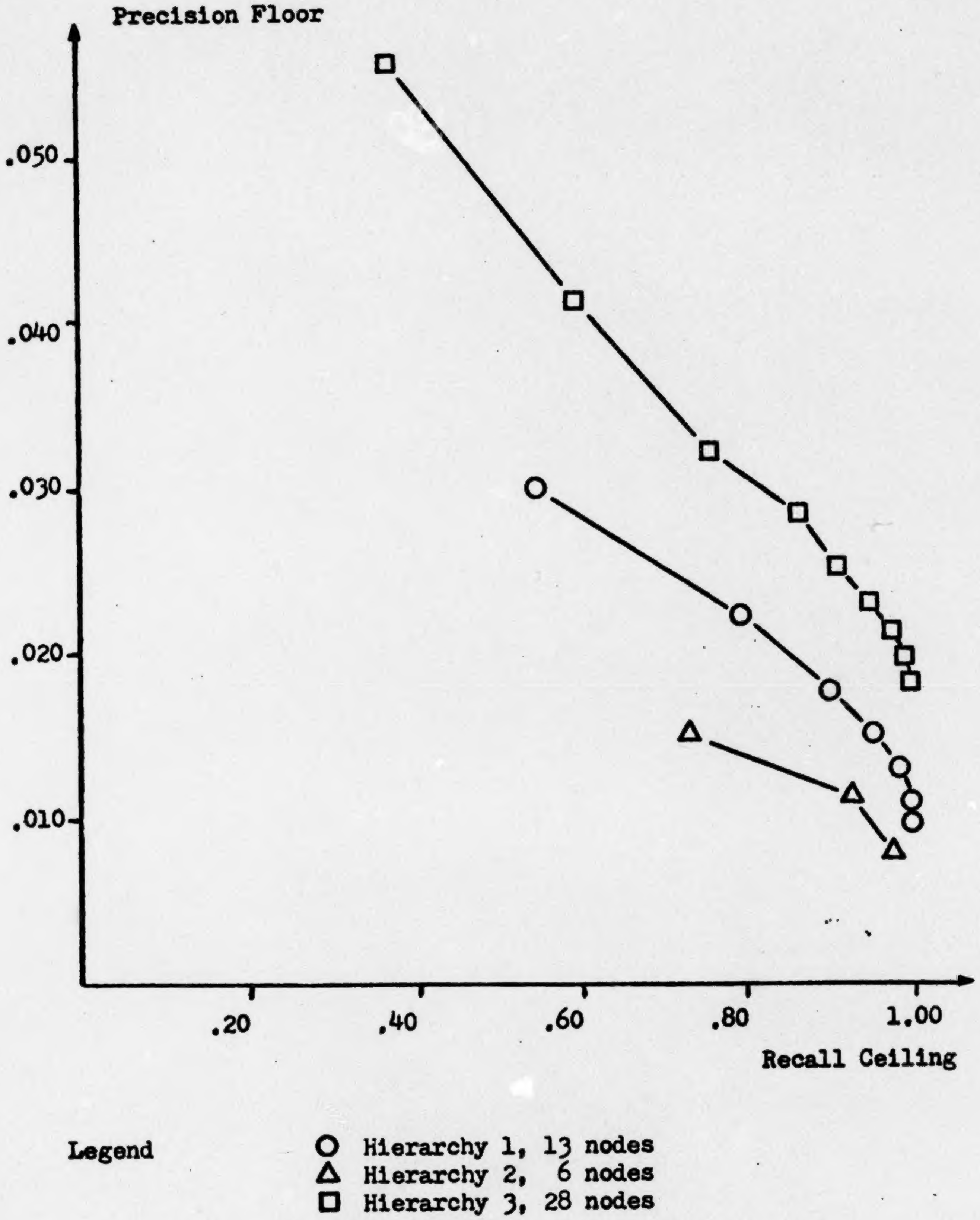
2) the single relevant cluster always ranks first when profiles are matched with the query vector.

In practice neither goal is achieved; therefore the best achievable performance curve lies beneath the ideal curve. Specifically, Table IV-1 shows that Dattola's classification algorithm resulted in 3 to 5 relevant nodes on level 1 and 5 to 9 relevant nodes on level 2. Under these conditions, best performance occurs if for each query i, the profiles are always ranked in a way that maximizes all partial sums

$$\sum_{k=1}^{j} \frac{r_{ik}}{c_{ik}} \qquad j = 1, 2, \ldots, m. \tag{IV-9}$$

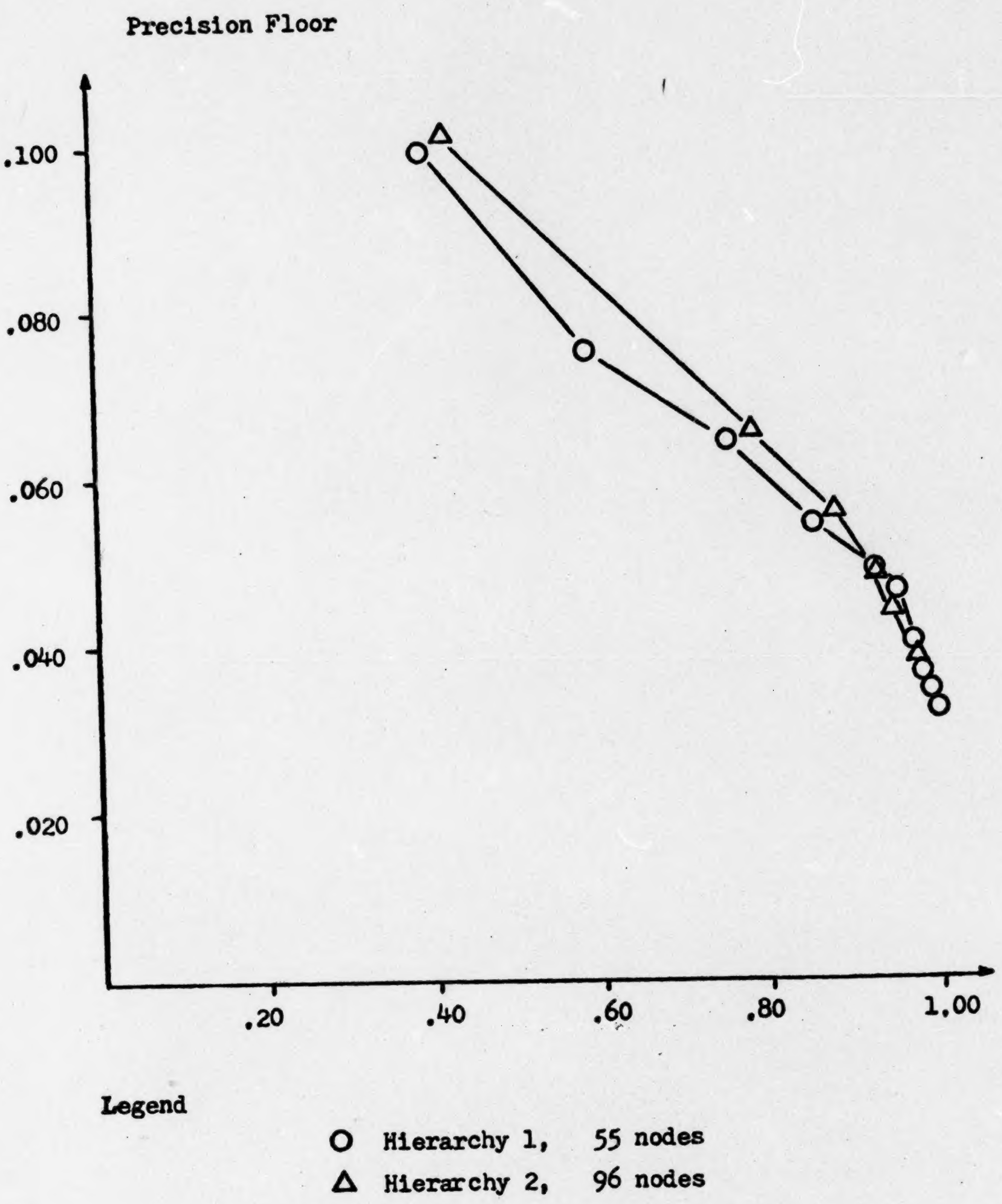
This ranking is best in the sense that the greatest number of relevant documents are retrieved for the fewest expanded nodes (least amount of work). Figures IV-8 and IV-9 show the best achievable performance for the first and second levels of the hierarchies used in this study.

Although cluster-oriented evaluation faces the drawbacks of incomplete searching and unsure relations between hierarchy levels, the method has several advantages. First, it is independent of expansion cutoffs and some parameters of various search strategies. Second, it accounts more accurately for system effort (hence, search cost) and leaves user effort as a secondary consideration. Third, it examines only a small part of the retrieval process rather than attempting to measure effects across an entire search. Presumably, the latter technique obscures some experimental effects in its across-the-board measurements. Finally, cluster evaluation is quite economical compared to actual searches. Both evaluation methods are used where appropriate so that conclusions are drawn with a substantial degree of confidence.



Best Achieveable Performance Curves--Level 1

Figure IV-8



Best Achieveable Performance Curves--Level 2

Figure IV-9

References

1. D. Williamson, R. Williamson, M. Lesk, The Cornell Implementation of the SMART System, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

2. C. W. Cleverdon, J. Mills, Factors Determining the Performance of Indexing Systems, Volume 1-2, Aslib Cranfield Research Project, 1966.

3. R. Dattola, Forthcoming Doctoral Thesis, Department of Computer Science, Cornell University.

4. G. Salton, The SMART Retrieval System--Experiments in Automatic Document Processing, Prentice Hall, Englewood Cliffs, N. J., 1971.

5. G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill, Inc., New York, 1968.

6. G. Salton, The Evaluation of Computer-Based Information Retrieval Systems, Proceeding of the FID Congress, 1965.

7. R. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Report ISR-16 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

Chapter V

Profile Experiments

1. Introduction

The previous chapters provide the background for the experiments in this and succeeding chapters. Previous chapters contain discussions on the structure and use of a clustered file, the basic profile definitions, search methods, storage organization, updating techniques, and other areas. The experimental environment, description of the collections and hierarchies, and the evaluation methods were also covered.

The present chapter presents the results of an extensive set of experiments focused on profile definition. Particular attention is given to:

a) the performance of the standard profiles (P_1,P_2,P_3);
b) the effects of rank value weighting;
c) bias in search results;
d) profile length;
e) frequency considerations; and
f) tradeoffs among unweighted, partially weighted, or fully weighted terms.

For the most part, the work uses Hierarchy 1 and cluster-oriented evaluation (see Chapter IV). The most promising techniques are thoroughly tested using all hierarchies and both evaluation schemes. Final conclusions are based on the complete set of results.

A summary of the major conclusions includes the following.

a) Profiles superior to standard or rank value vectors can be made by using term weights based on frequency ranks (not rank values). The resulting vectors are free from correlation domination and other biases.

b) A large portion of the low weighted profile terms may be deleted without a large performance loss. In fact, deletion improves the performance of unweighted profiles.

c) Unweighted profiles give somewhat inferior search performance, but partial weighting schemes may suffice instead of fully weighted profiles.

A number of secondary conclusions related to cluster size, biased search results, and frequency considerations are brought forth also.

2. Standard Profile Performance

In order to review the standard profile definitions, consider a node whose crown is the document set $C = \left\{D_1, D_2, \ldots, D_n\right\}$. Then, the standard profiles are

$P_1 = D_1 \vee D_2 \vee \ldots \vee D_n$ where D_i is an unweighted vector,

$P_2 = D_1 + D_2 + \ldots + D_n$ where D_i is an unweighted vector, and

$P_3 = D_1 + D_2 + \ldots + D_n$ where D_i is a weighted vector.

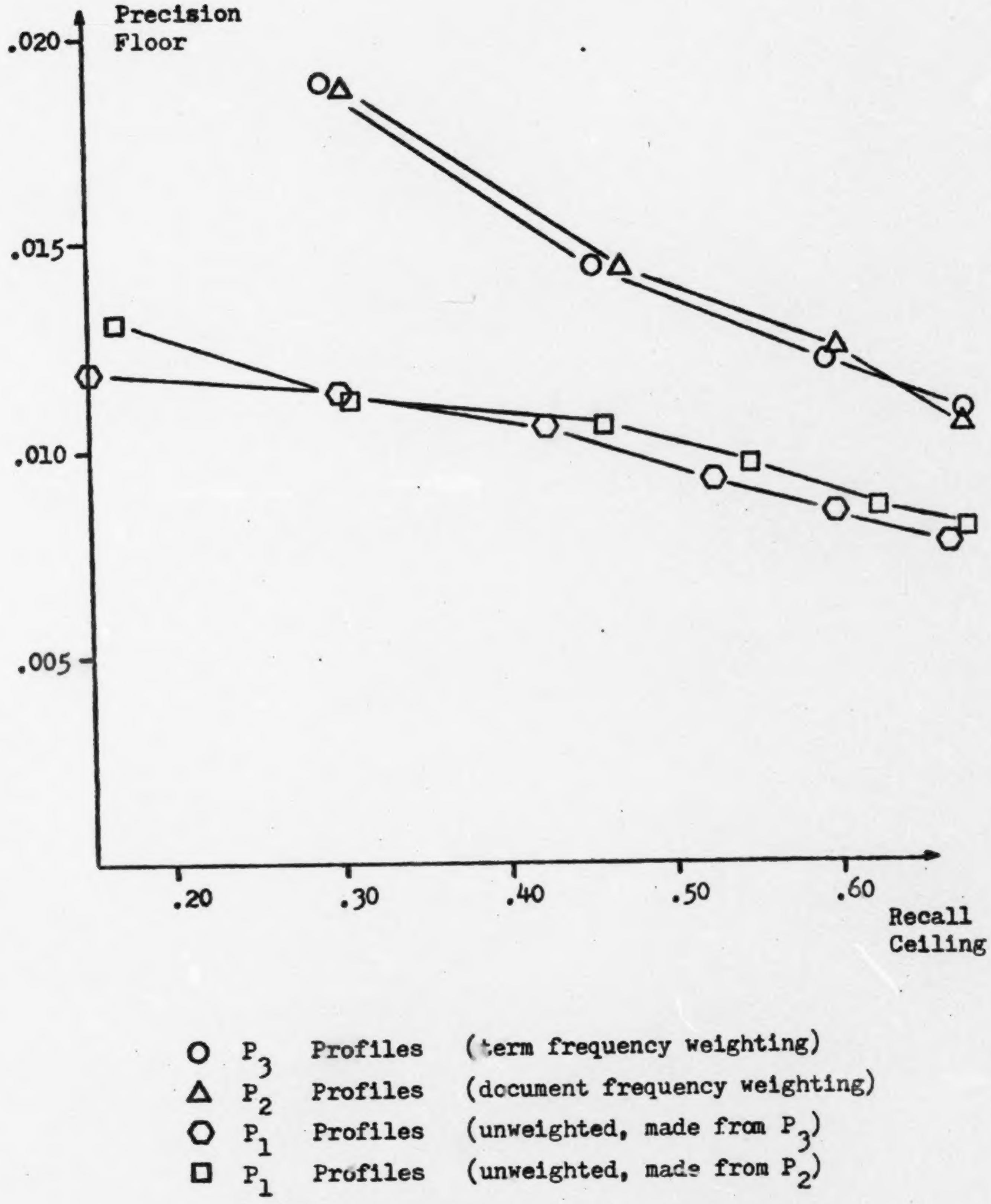
Terms in P_1 profiles are unweighted while those in P_2 and P_3 profiles are weighted according to document frequencies or total occurrence frequencies within C. In Chapter IV it was explained how each profile is obtained from the clustered Cranfield collection including the necessity of eliminating the lowest frequency terms. In order to describe other profile properties, the following concepts are introduced. The size of a profile P is the number of documents in its crown; its *length* is the number of index terms in its vector; and its *magnitude* $|P|$ is the square root of the sum of squares of its term weights. In the case of P_1 vectors, each term is assigned a unit weight. The properties of the standard profiles for Hierarchy 1 are given Table V-1.

Evaluation curves for the standard profiles are shown in Figures V-1 and V-2. At least three observations can be made. First *weighted profiles perform significantly better than unweighted ones* ($P_3 > P_1$, $P_2 > P_1$). Later examination shows that the results for unweighted vectors are biased so that small clusters unfairly achieve high ranks, regardless of their relevancy. Some of this bias can be removed and performance improves considerably. *A second observation is that term weights based on document frequency appear equivalent, if not slightly superior, to weights based on total term occurrence (using within document frequencies*), i.e. $P_2 > P_3$. This is a surprising and pleasing result since it indicates that an existing document collection without weights can be clustered and searched without performance loss due to profiles. If a large performance difference had been observed, an unweighted collection would have to be re-indexed with weights in order to obtain maximum benefit from the clustered organization. *The final observation is that of a slight performance advantage for the shorter P_1 vectors over the longer ones*. The effects of vector length in unweighted profiles is discussed in Section V.7. Actually the standard profiles differ in so many ways it is impossible to draw conclusions from these tests. The curves are presented as a reference for later experiments involving fewer variables.

Property	Nodes on Level 1	Nodes on Level 2
Number of profiles	13	55
Average size	115	28
P_3 Profiles (term frequency weighting)		
Average length	812	323
Range of lengths	438-1302	120-692
Average magnitude	590	162
Range of magnitudes	340-971	74-304
P_2 Profiles (document frequency weighting)		
Average length	722	266
Range of lengths	397-1175	93-580
Average magnitude	291	84
Range of magnitudes	169-477	37-145
P_1 Profiles (unweighted, made from P_3)		
Average length	812	323
Range of lengths	438-1302	120-692
Average magnitude	28	44
Range of magnitudes	22-36	11-26
P_1 Profiles (unweighted, made from P_2)		
Average length	722	266
Range of lengths	397-1175	93-580
Average magnitude	27	16
Range of magnitudes	20-33	10-24

Properties of P_1, P_2, P_3 Profiles for Hierarchy 1

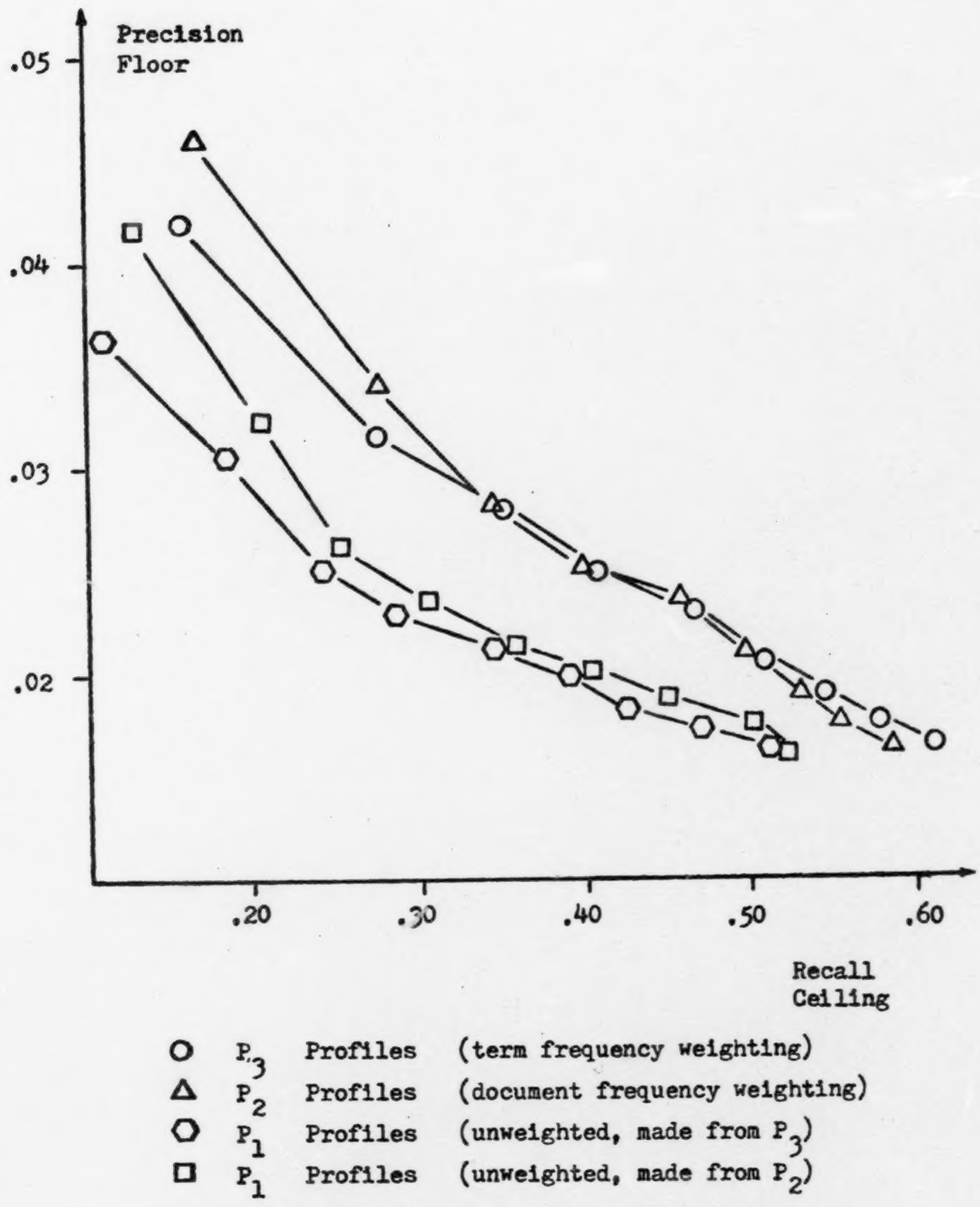
Table V-1



Evaluation of the Standard Profile Definitions

Hierarchy 1, Level 1

Figure V-1



Evaluation of the Standard Profile Definitions

Hierarchy 1, Level 2

Figure V-2

Because many curves similar to Figures V-1 and V-2 are presented, a few remarks about their characteristics are in order. First, points are plotted at cluster cutoffs in all cases so that the i^{th} point represents the precision floor and recall ceiling obtained if the search expands i clusters. Roughly speaking, the same amount of system work (e.g., number of disk fetches) can be associated with the first, second, third, etc., points on all curves for a given hierarchy level. Second, the PF scale varies considerably between levels while the RC scale is the same. This is due to the substantial difference in the size of the profiles on various levels and the dependency of PF on profile size. Third, more performance differences are generally observed on upper hierarchy levels. This is caused by the nature of these vectors--longer, more extreme weights, greater magnitudes, etc.

3. Rank Value Profiles

Rank value profiles derive term weights from frequency ranks rather than frequency counts. Given a P_2 or P_3 vector, its terms are ordered by decreasing frequency and re-weighted by assigning them rank values. A rank value is the difference between a base value and the position of the term in the frequency ranking. Chapter III illustrates rank value profiles and points out their differences from standard profiles; namely

a) all vectors have the same high weight rather than the same low weight and

b) the range of term weights is considerably reduced since weights are derived from frequency ranks.

The following experiments examine the selection of a base value and these differences. The results indicate superior performance can be achieved through rank value weighting. However, improvements are due to changes in the physical properties of the vectors rather than to factors intrinsic to the rank values themselves.

A. Base Value Selection

Suppose a rank value profile P has k index terms with frequency ranks from 1 to $r \leq k$ (terms with the same frequency share the same rank). If the base value is $b > r$, term weights range from $w_o = b-r$ to $w_a = b-1$. The quantity w_o is the weight origin (lowest value), w_a is the weight apex (highest value), and $w_a - w_o = r$ is the weight range. As mentioned above, keeping the base value constant for all profiles assures that all vectors have the same apex. This contrasts with the standard profiles which all have the same origin.

In Doyle's work (1), the major criterion in base value selection is assurance of a positive weight origin in all profiles. However, the base value influences cosine correlations and search results and therefore should be chosen carefully. To illustrate, consider a rank value profile $P = (P_1, P_2, \ldots, P_v)$ with a base value b such that $w_o = 1$. Increasing the base value to $b' = b + a$ $(a > o)$ is equivalent to increasing all term weights by a constant and forming

$$P' = P + A \text{ where } A = (a_1, a_2, \ldots, a_v),$$

$$a_i = \begin{cases} a = b' - b \text{ if } p_i \neq 0 \\ 0 \text{ if } p_i = 0 \end{cases} \qquad \text{(V-1)}$$

The addition vector A is actually an unweighted profile whose unit weight is a. Correlations involving P' can be expressed as follows:

$$\begin{aligned}\cos(Q,P') &= \cos(Q,P+A)\\ &= \frac{|P|}{|P+A|}\cos(Q,P) + \frac{|A|}{|P+A|}\cos(Q,A)\\ &= \Big\{\alpha\cos(Q,P) + \beta\cos(Q,A)\Big\}\\ &\qquad\Big\{1 - 2\,\alpha\beta\,\big[1-\cos(P,A)\big]\Big\} \qquad \text{(V-2)}\\ &\approx \alpha\cos(Q,P) + \beta\cos(Q,A)\end{aligned}$$

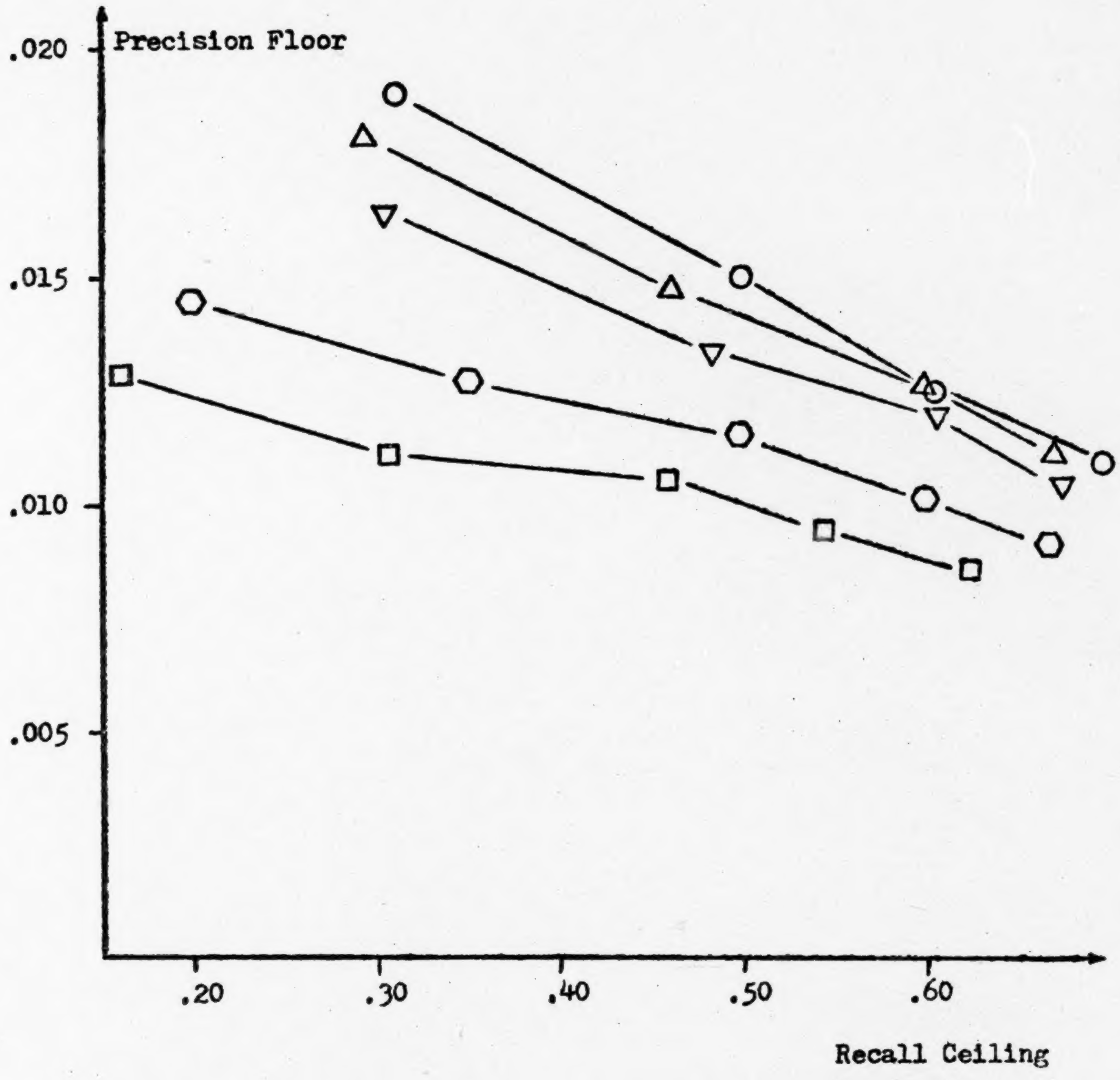
where $\alpha = \frac{|P|}{|P + A|}$ $\beta = 1 - \alpha = \frac{|A|}{|P + A|}$

This equation indicates that the total correlation is approximated by a linear combination of two other correlations--one from the original weighted profile P and the other from the unweighted profile A. Note that α and β depend only on $|P|$ and $a = b' - b$. Further, as the base value increases, $w_o \rightarrow \infty$, $w_a \rightarrow \infty$, $\alpha \rightarrow 0$, and $\beta \rightarrow 1$. As a result, the unweighted correlation dominates the total and performance approaches that of unweighted profiles.

The effect of increasing the base value can also be viewed as making terms less distinguishable during the correlation process. For example, the terms of the profile P = (2, 1) using b = 3 contribute to correlations in the ratio 2:1. That is, a match on one term is worth twice as much as a match on the other. Raising the base value to b' = 11 yields P' = (10, 9) whose terms contribute in the ratio 10:9. The relative importance of terms is reduced so that correlations differ only slightly depending on which term is actually matched. The same effect occurs in large vectors also, namely an increase in base value "smears" the importance of the weights assigned to individual terms. In the profiles for these experiments, term weights represent frequencies; using a low base value maintains frequency distinctions while a high value decreases their importance.

Figures V-3 and V-4 compare search performance in Hierarchy 1 for several sets of rank value profiles made from P_2 vectors (document frequency weighting). The results are generally as predicted by equation V-2, namely decreasing performance with increasing base value. This supports the idea of maintaining the distinctions apparent in the original term weights to whatever extent possible. The single exception to these conclusions occurs in Figure V-3 for the lowest base value (66). One explanation for $P_{b = 86} > P_{b = 66}$ is that too small a base value places unwarranted importance on frequency rank as a retrieval indicator. However, other data shows that the $P_{b = 66}$ performance is strongly influenced by a single, large cluster which nearly always ranks high regardless of its relevancy. Some evidence of this situation lies in the fact that the abscissa (RC) values are nearly identical for both curves while their ordinate (PF) values differ markedly because of cluster size (see equation IV-5). It is not the case that an equal number of relevant documents could not be retrieved, but that they are recovered from clusters of vastly different sizes. The exact nature of this bias is discussed further in Section V.4.

The evidence shows that rank value profiles perform better when they rely on a small base value rather than a large base value. Since the base value is selected prior to profile construction, it is difficult to determine a value which is low, but not so low as to jeopardize performance

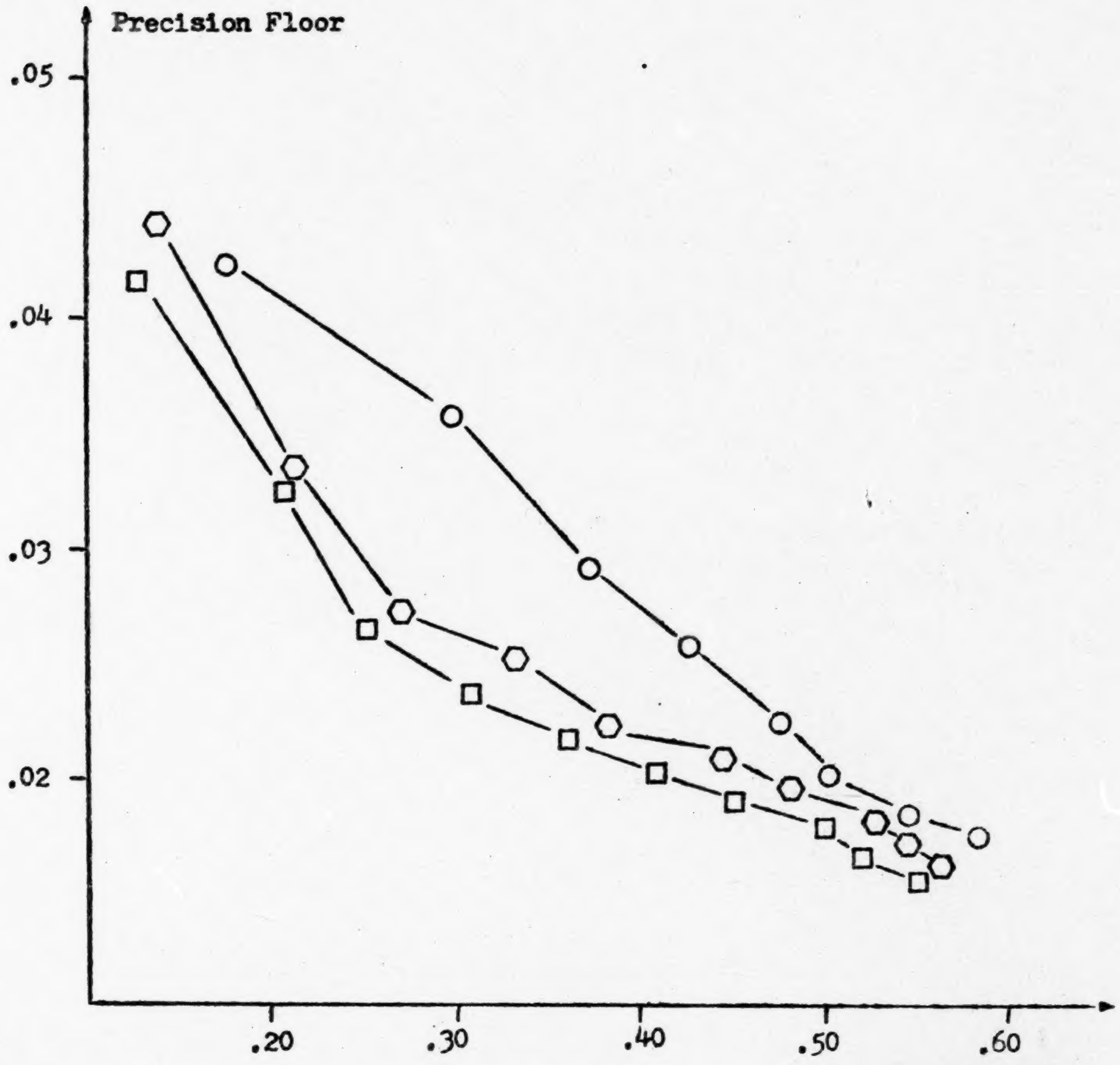


Symbol	Base Value	Apex	Lowest Origin
▽	66	65	2
○	86	85	22
△	100	99	36
⬡	226	225	162
□	∞	(unweighted)	

Search Performance as a Function of Base Value

Rank Value P_2 Profiles, Hierarchy 1, Level 1

Figure V-3



Symbol	Base Value	Apex	Lowest Origin
○	26	25	2
⬡	100	99	76
□	∞	(unweighted)	

Search Performance as a Function of Base Value

Rank Value P_2 Profiles, Hierarchy 1, Level 2

Figure V-4

as in Figure V-3 (b = 66). Fortunately, once the profiles are made, they can be adjusted via Equation V-1 to any desired base value. The experiments in the following sub-section show how to eliminate the entire problem of base value selection.

B. Weight Origins and Apexes

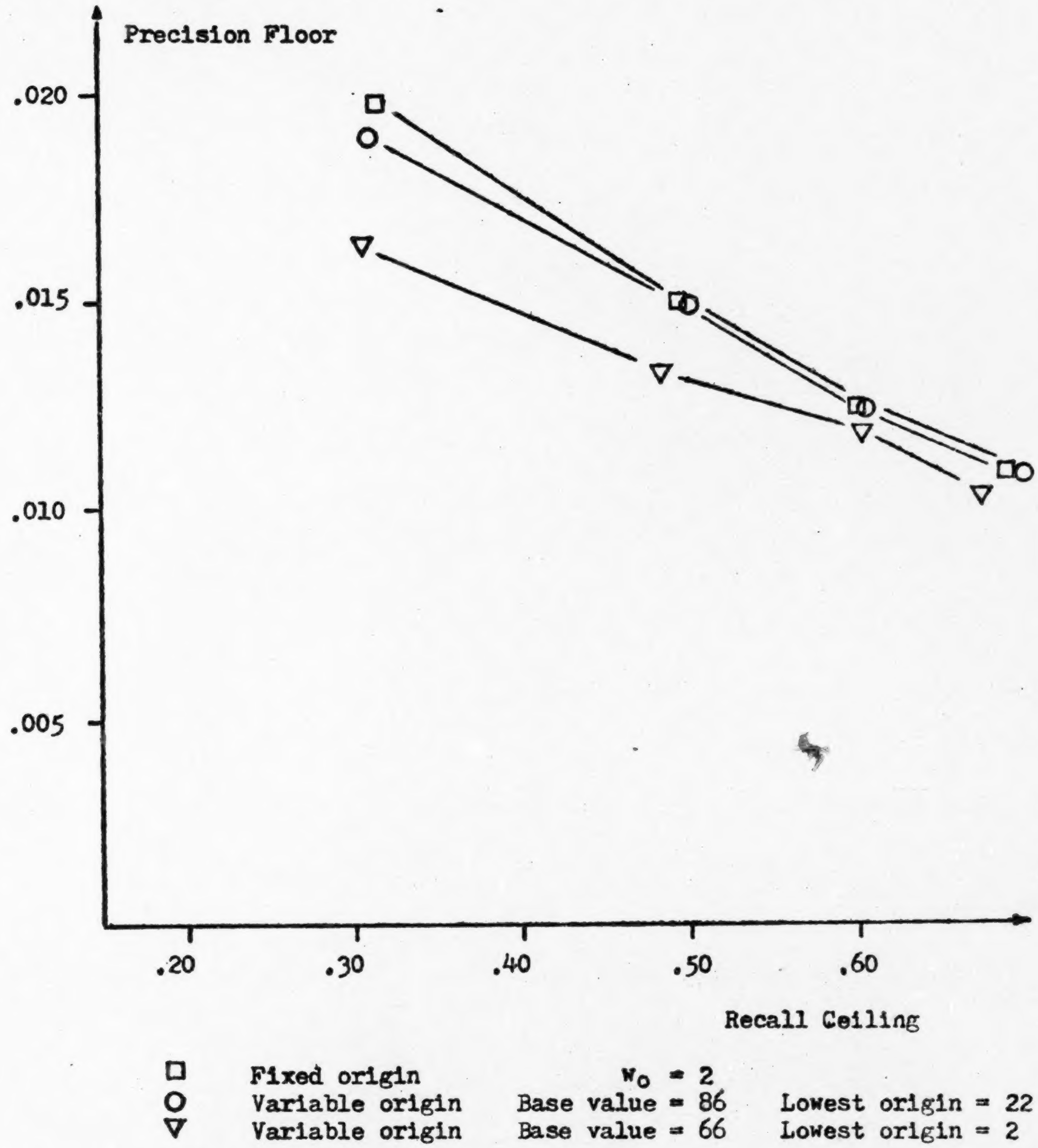
Using the same base value throughout a hierarchy or level causes all profiles to have the same weight apex while their origins vary. For example if b = 21, the hierarchy might contain these profiles:

$P = (18,20,19)$ $\quad w_o = 18,\ w_a = 20$

$P' = (20,16,18,15,12,19,16,17,13,14)$ $\quad w_o = 12,\ w_a = 20$

The previous experiment supports, to some extent, the notion of maintaining maximum differentiation among profile terms in rank value profiles by keeping the base value (and hence weight origins) low. A logical extension of this idea is to artifically reduce the weight origins for all vectors to the same low value. Note that this does not produce a standard P_2 or P_3 vector since profile term weights are still based on frequency ranks.

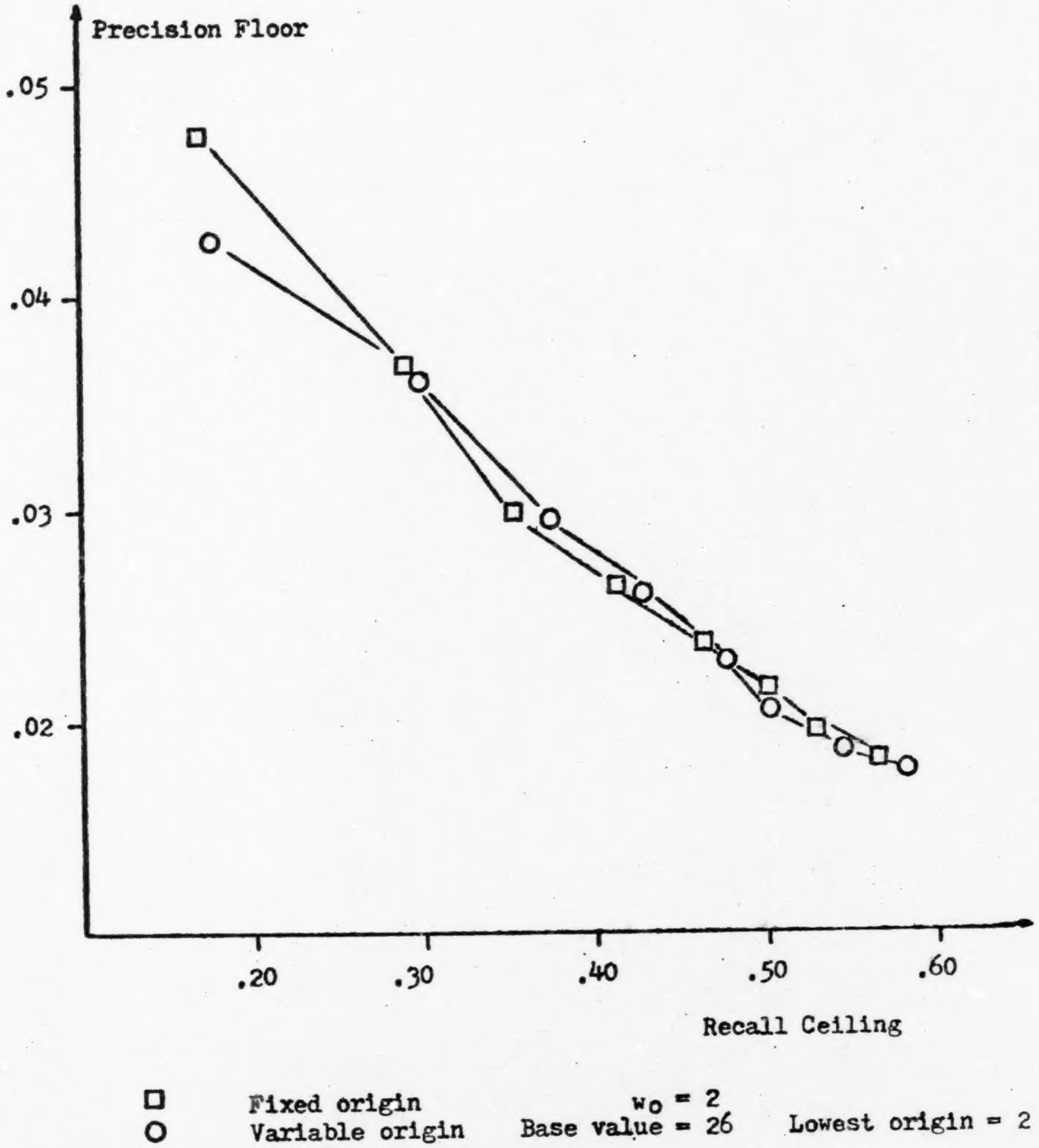
Figures V-5 and V-6 compare the performance of rank value profiles with variable weight origins (fixed apex) and similar vectors with a fixed weight origin (variable apexes). The experimental profiles are constructed from original P_2-type vectors (document frequency weighting) and are designed so that the lowest origin is the same in all tests. The best previous curves are included also. The figures show no significant difference between good profiles with variable origins and profiles with



Comparison of Fixed and Variable Weight Origins

Rank Value V_2 Profiles, Hierarchy 1, Level 1

Figure V-5



Comparison of Fixed and Variable Weight Origins

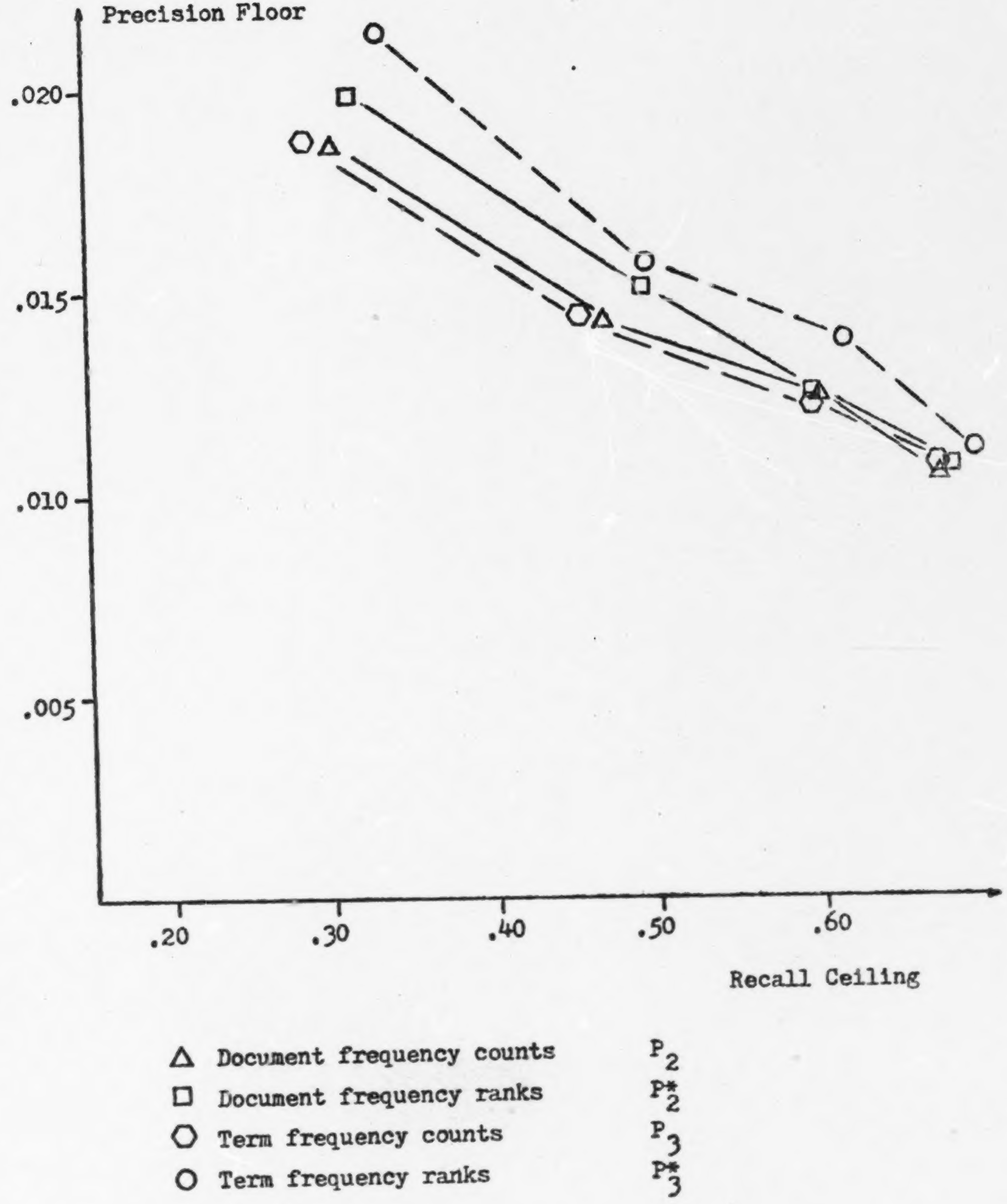
Rank Value P_2 Profiles, Hierarchy 1, Level 2

Figure V-6

a low fixed origin. This is not unexpected in level 2 vectors, where the change in origin produce only small changes in term weights. However, most level 1 profiles have their term weights reduced by a considerable amount, yielding vectors whose correlations are more sensitive to individual term weights (see Section V.2.A.). As a result, the size bias noted earlier is removed and performance improves. The importance of these tests is that they show rank value profiles with a fixed, minimal weight origin provide equivalent or better performance than similar profiles constructed using an optimal base value (variable origins). Consequently, base value selection need not be considered in profile construction. The new construction process simply sorts the index terms of an initial vector (P_2 or P_3) in increasing frequency order and assigns weights equal to ranks in the sorted sequence. These vectors are denoted by P_2^* or P_3^*, depending on the initial vector.

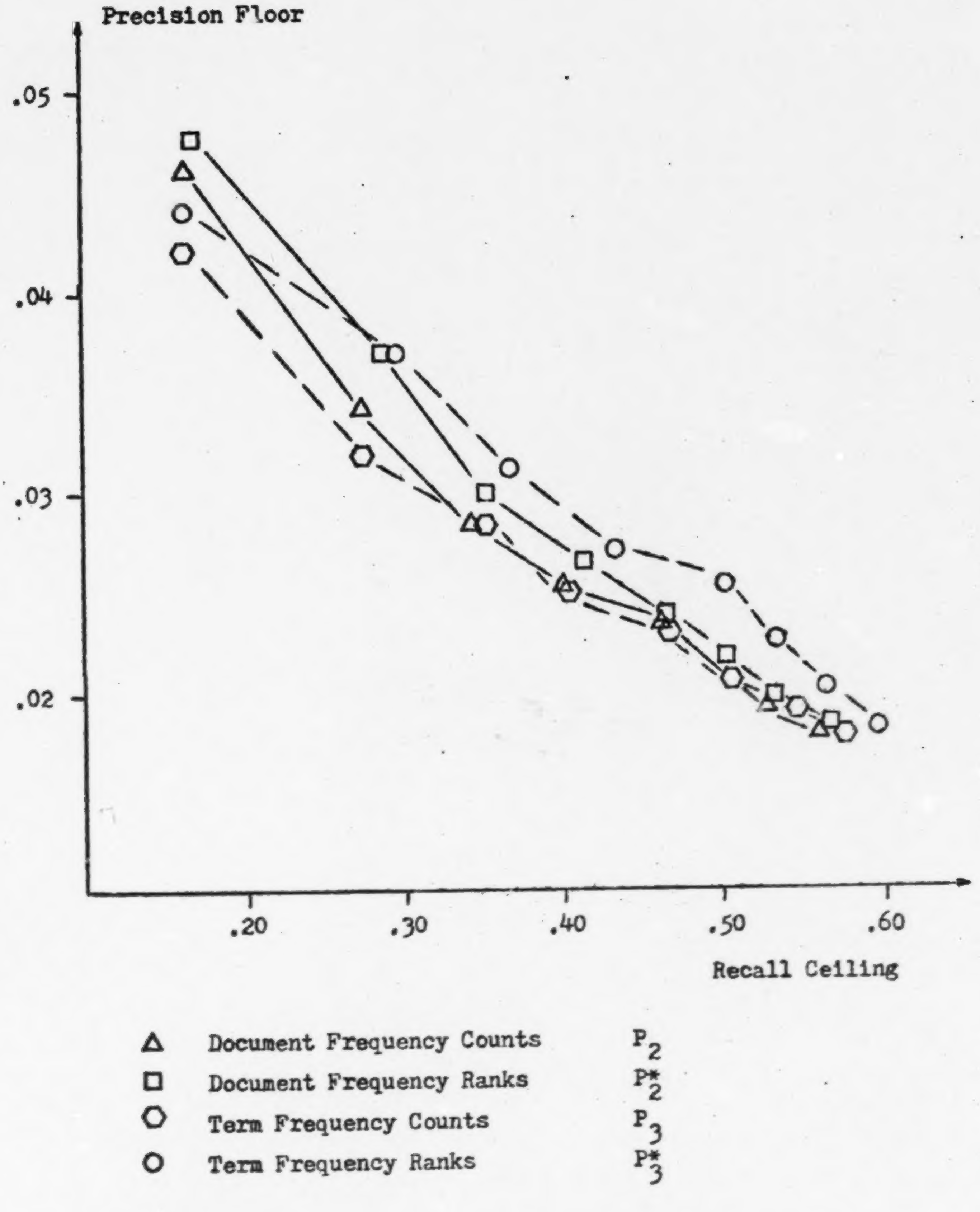
C. Weight Range

The starting points for rank value profiles are standard vectors (P_2 or P_3) whose term weights are frequency counts. Given such rank value profiles, the previous experiments suggest reducing their weight origins to a minimal constant in order to improve performance (P_2^* or P_3^* vectors)). The difference between these final profiles and the standard profiles is simply that term weights are ranks of document frequencies rather than frequency counts. Figures V-7 and V-8 compare the effectiveness of rank weighting and count weighting for both document and term frequencies. In all cases P_2 and P_2^* curves are connected by solid lines while P_3 and P_3^* curves are connected with broken lines.



Comparison of Profile Term Weights Based on Frequency Counts and Frequency Ranks--Hierarchy 1, Level 1

Figure V-7



Comparison of Profile Term Weights Based on Frequency

Counts and Frequency Ranks--Hierarchy 1, Level 2

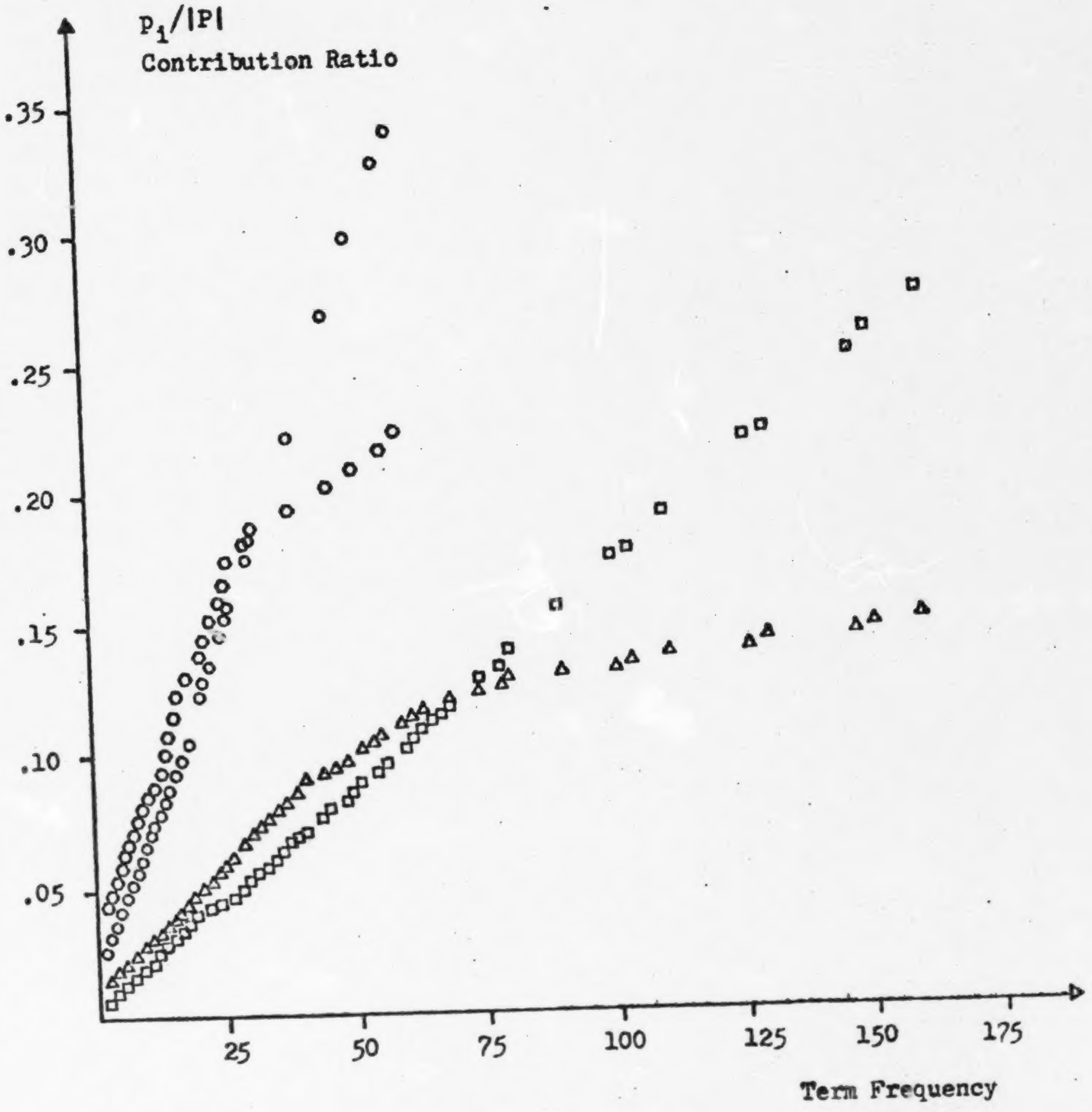
Figure V-8

In order to interpret the curves, first suppose that a fixed decision is made to weight profile terms either according to frequency counts or ranks. Then the evaluation shows no consensus on a preference for document or term frequencies, that is, $P_2 \approx P_3$ and $P^*_2 \approx P^*_3$. This result also supports the findings of Section V.2. In a similar manner, consider a fixed decision to base all profile term weights either on document or term frequencies. In this case, the tests suggest that weights based on ranks are superior to those based on counts, that is, $P^*_2 > P_2$ and $P^*_3 > P_3$. The major reasons for this performance improvement are the altered profile characteristics, specifically a reduced weight range and its effect on the cosine correlation, as shown below.

The magnitude of a profile vector and the presence or absence of a few term matches may greatly influence the final cosine correlation. Given a profile $P = (p_1, p_2, \ldots, p_v)$, where p_i is the weight of term i, and a similar query $Q = (q_1, q_2, \ldots, q_v)$ matching terms with weights p_i and q_i contribute, to the total correlation, the amount:

$$\text{CONTRIBUTION} = \left(\frac{q_i}{|Q|}\right)\left(\frac{p_i}{|P|}\right) = \left(\frac{q_i}{\left[\sum q_j^2\right]^{\frac{1}{2}}}\right)\left(\frac{p_i}{\left[\sum p_j^2\right]^{\frac{1}{2}}}\right) \quad \text{(V-3)}$$

For any particular query, the values of $q_i/|Q|$ are fixed and variations in contributions are due to $p_i/|P|$. Figure V-9 is a plot of the contribution ratio, $p_i/|P|$, for all unique term weights found in typical profiles from Hierarchy 1. Two versions of each vector are shown, one having weights based on frequency counts and the other using frequency ranks. This type of curve is called a correlation contribution curve and is used frequently in this study. The curves for frequency counts, show that about 3% of all



Symbol	Level	Node	Profile Type	Weighting
□	1	5	P_3	Term Frequency Counts
△	1	5	P^*_3	Term Frequency Ranks
○	2	14	P_3	Term Frequency Counts
⬡	2	14	P^*_3	Term Frequency Ranks

Cosine Correlation Contribution Ratios--Hierarchy 1

Figure V-9

terms have very large contribution ratios and control retrieval in the sense that their matches practically guarantee expansion of the corresponding cluster. This control is due to the fact that

a) large term weights increase $|P|$ a great deal and result in large correlation contributions;

b) small term weights change $|P|$ very little and are relegated to small contributions; and

c) high contributions are obtained at the expense of low ones since their total is bounded $(\text{MAX} \left\{ \text{COS}(P,Q) \right\} = 1)$.

Without high weight matches, a considerable number of other terms must match in order to expand a cluster. This, however, is rather uncommon since queries generally contain only a few index terms (an average of 9 in the Cranfield collection). This power of a few high weight profile terms to influence correlation and, hence, search outcome is called correlation dominance. The contribution curves for profiles with weights based on frequency ranks show much less domination. In these vectors, the range of weights is much smaller, $|P|$ is smaller, and the correlation ratios are more evenly distributed. Frequent terms are still more important than non-frequent terms, but they no longer dominate since it is easier for a number of other terms to influence cluster expansion. The reduced correlation domination is the factor leading to the performance improvement noted in Figures V-7 and V-8. Although frequency ranks are used here, later experiments show that domination can be reduced by other methods also. The ranking scheme is simply convenient and maintains weights which are non-decreasing with frequency.

Another implication of these results is that the importance of an index term for retrieval does not increase linearly with frequency, but in a more gradual way. While previous experiments show the validity of increasing frequency distinctions among terms, by shifting to minimal weight origins, it is clear that extreme distinctions (domination) must be avoided. Term weights based on frequency ranks benefit because they meet both criteria. For these reasons, the P_2^* and P_3^* profiles provide the best performance encountered in this research.

Since the P_2^* and P_3^* profiles appear throughout this study, it is appropriate to give a complete example of their construction. Figure V-10 contains such an example and includes a comparison with the standard P_2 and P_3 vectors. The sample data is the same as that used in Chapter III for similar purposes. As shown in the figure, the starting point is a standard profile whose term weights are simple frequency counts. First, all terms are ranked in increasing frequency order; that is, the least frequent term receives rank 1, etc. This contrasts with Doyle's scheme of using decreasing order. Note that terms of equal frequencies share the same rank. Second, the final profile is made by replacing each original term weight with the term rank established in the previous step. As discussed earlier, this process reduces the range of weights in the profile, the vector magnitude, and the range of correlation contributions from matching terms. This is particularly true for P_3^* vectors. These factors decrease the amount of correlation domination and lead to the performance observed earlier.

D. Summary

This section consists of an investigation of rank value profiles

P_2^* Profile (Weights based on document frequency ranks)		Magnitude
Original Profile	$P_2 = (1,0,2,0,0,5,3,0,0,2)$	$\sqrt{43}$
Frequency ranks	1 2 4 3 2	
P_2^* Profile	$P_2^* = (1,0,2,0,0,4,3,0,0,2)$	$\sqrt{34}$

Contributions of Term Matches to the Total Cosine Correlation

Contribution Vector	Matching Term 1	2	3	4	5	6	7	8	9	10
$P_2/\lvert P_2\rvert$	.15	0	.30	0	0	.76	.46	0	0	.30
$P_2^*/\lvert P_2^*\rvert$	.17	0	.34	0	0	.69	.51	0	0	.34

P_3^* Profile (Weights based on term frequency ranks)		Magnitude
Original Profile	$P_3 = (1,0,5,0,0,10,4,0,0,5)$	$\sqrt{167}$
Frequency Ranks	1 3 4 2 3	
P_3^* Profile	$P_3^* = (1,0,3,0,0,4,2,0,0,3)$	$\sqrt{38}$

Contributions of Term Matches to the Total Cosine Correlation

Contribution Vector	Matching Term 1	2	3	4	5	6	7	8	9	10
$P_3/\lvert P_3\rvert$	.08	0	.39	0	0	.77	.31	0	0	.39
$P_3^*/\lvert P_3^*\rvert$	.16	0	.48	0	0	.64	.32	0	0	.48

Construction of P_2^* and P_3^* Profiles

Figure V-10

and their differences from vectors considered previously. The results include the following points.

a) Low, but not minimal base values are preferable to large base values, an optimal choice being difficult to find.

b) The problem of base value selection can be eliminated by using a fixed, minimal weight origin for each profile.

c) Term weights based on frequency ranks are superior to weights based on frequency counts.

These properties are present in the P_2^* and P_3^* vectors which generally yield performance superior to either the standard profiles or rank value profiles. The modified profiles are actually variations of the latter types. Their only difference from standard vectors is the use of term weights based on frequency ranks. The change from a rank value profile is the use of a fixed, minimal weight origin for each vector rather than a global base value. In any case, all remaining experiments consider the use and characteristics of P_2^* or P_3^* vectors unless explicity stated otherwise.

4. Search Bias

A. An Algorithm for Detecting Bias

The cluster search process consists of matching a query with all profiles on the first level of the hierarchy, ranking them in correlation order, and selecting several nodes for expansion. The process is repeated until document vectors are reached. There is a natural curiosity about the properties of profiles which occupy the initial portions of the ranking on each level and thereby become expanded. Specifically, it is desirable to determine whether these profiles have consistent properties of length, size, magnitude, etc. If a damaging bias is detected, then steps should be taken to correct it. For example, the discussion of the experiments on base value selection mentions a bias toward large clusters in an informal manner. This section formalizes this concept and develops an analytical procedure for detecting search performance biased by a particular profile property. Using this technique on various types of profiles reveals the negative influence of a bias in unweighted and certain types of rank value profiles.

The analysis procedure starts with data from a given search--so many relevant and non-relevant clusters ranked above a chosen cutoff on each hierarchy level--and calculates the expected participation of each profile in achieving this performance. It then examines the actual participation of each profile and notes deviations from expected values. Patterns of large deviations of behavior denote bias. In general, good recall ceiling--precision floor resulst are accompanied by little or no bias. As with cluster-oriented evaluation (RC-PF), the bias analysis considers each hierarchy level separately; the arguments for and against this approach are given in Section IV.4.

For a detailed description of the analysis method, consider a single hierarchy level with K profiles $\left\{P_1, P_2, \ldots, P_K\right\}$ and a collection of J requests. Let there be a total of $\sum R_i$ relevant clusters and $\sum N_i$ non-relevant clusters where

a) R_i is the number of requests for which P_i is relevant and

b) N_i is the number of requests for which P_i is non-relevant.

Consequently, $I = \lceil \sum R_i/J \rceil$ is approximately the average number of relevant clusters per request so that if the expansion cutoff I is used, then on the average, all relevant profiles could occupy rank positions 1, 2, ..., I. In actual tests, relevant profiles ranking I or above are said to perform well while non-relevant profiles ranking I or above are said to perform poorly. From an actual search then, the following data is collected:

c) r_i, the number of queries for which P_i is relevant and ranks I or above and

d) n_i, the number of queries for which P_i is non-relevant and ranks I or above.

Looking at a specific profile P_i, the ratio r_i/R_i is its relative frequency of good performance while n_i/N_i is its relative frequency of poor performance. Over the entire set of profiles, values for these ratios which differ markedly from expected values may indicate biased results. Figure V-11a contains sample data for 5 profiles and 20 requests. For example, there is a total of $\sum R_i = 40$ occurrences of relevant clusters distributed as 2, 5, 7, 12, and 14 occurrences among individual clusters. Therefore a typical request has $I = \lceil \sum R_i/J \rceil = 40/20 = 2$ relevant clusters. This data is fixed and unchangeable for the given collections. In an actual search, suppose $\sum r_i = 29$ relevant profiles rank 1 or 2 with these occurrences being distributed 1, 3, 5, 9, and 11 among the individuals. Taking simple ratios r_i/R_i gives the indicated frequencies of good performance. A similar explanation applies to the non-relevant items. Because of the way the data is listed, it is easy to observe some correlation between cluster size and either performance ratio n_i/N_i or r_i/R_i. Without knowing how much these ratios differ from their expected values, it is impossible to say whether the search results are biased. The following analysis examines this situation more closely.

Overall, any particular profile P_i is non-relevant for $N_i/\sum N_j$ of all queries. Furthermore, P_i accounts for $n_i/\sum n_j$ of all non-relevant profiles ranked in the first I positions. There is no a priori reason for P_i to be more prevalent in these positions than any other non-relevant profile, so the expected value of n_i/N_i is:

$$E_n = E\left\{n_i/N_i\right\} = \frac{\sum n_j}{\sum N_j} \qquad \text{(V-4)}$$

Note the expression is independent of the profile under consideration. A similar calculation for the ratio r_i/R_i yields

$$E_r = E\left\{r_i/R_i\right\} = \frac{\sum r_j}{\sum R_j} \qquad \text{(V-5)}$$

The latter expression represents the true expected value only if all relevant clusters contain the same ratio of relevant to non-relevant documents for all queries. In practice, cluster sizes and the number of relevant documents in them vary a great deal and the above condition does not hold. Consequently, the conclusions of forthcoming tests are based primarily on the behavior of non-relevant profiles (n_i/N_i) and supported by the behavior of relevant profiles (r_i/R_i).

In either case, the important quantities are the deviations of experimental values from expected values, namely $(n_i/N_i)-E_n$ and $(r_i/R_i)-E_r$. Figure V-11b shows these deviations for each of the

Number of profiles, K = 5

Number of queries, J = 20

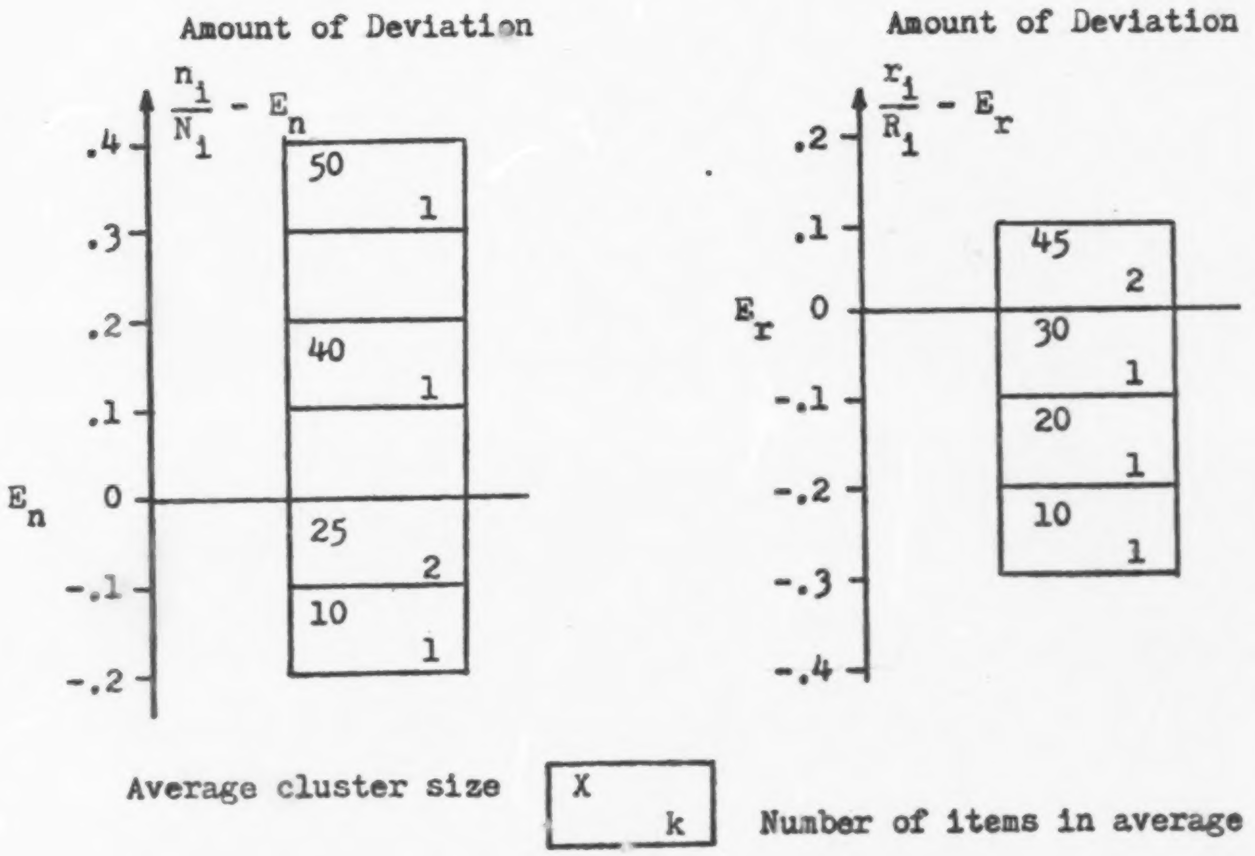
Property	Profiles P_1	 P_2	 P_3	 P_4	 P_5	Totals
Cluster size	10	20	30	40	50	150
R_i	2	5	7	12	14	$\sum R_i = 40$
N_i	18	15	13	8	6	$\sum N_i = 60$
$I = \lceil \sum R_i / J \rceil = 2$						
r_i	1	3	5	9	11	$\sum r_i = 29$
n_i	1	2	2	3	3	$\sum n_i = 11$
n_i/N_i	.06	.13	.15	.37	.50	
r_i/R_i	.50	.60	.72	.75	.79	

a) Sample Data Related to Biased Searches

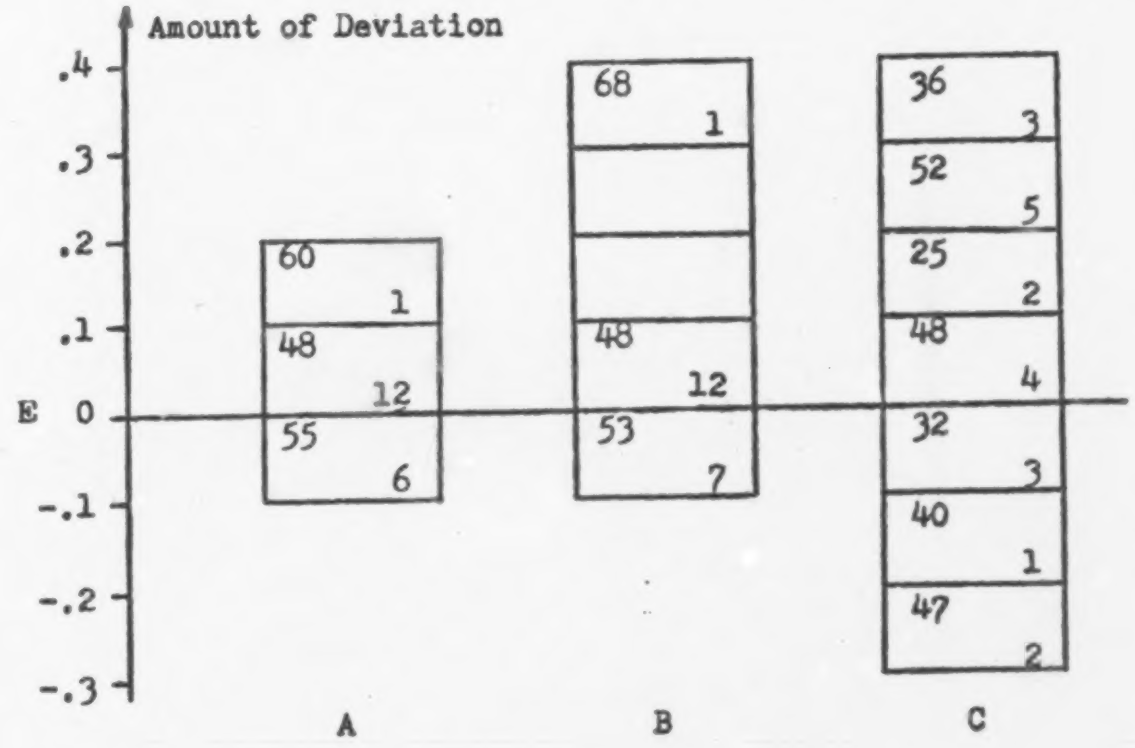
Property	Profiles P_1	 P_2	 P_3	 P_4	 P_5
$E_n = \sum n_i / \sum N_i$	.18	.18	.18	.18	.18
$n_i/N_i - E_n$	-.12	-.05	-.03	.19	.32
$E_r = \sum r_i / \sum R_i$	.72	.72	.72	.72	.72
$r_i/R_i - E_r$	-.22	-.12	0	.03	.07

b) Deviations of Performance from Expected Values

Figure V-11



c) Relating Cluster Size and Performance Deviations



d) Examples of Unbiased Search Results

Figure V-11 Continued

profiles in the example. Figure V-11c provides a way of observing search results biased with respect to cluster size because it lists the average size of all clusters whose profile behavior falls within various deviation intervals. The second number in each square is the number of items used in the average. Thus in the deviation interval $[E_n-.1, E_n)$, profiles P_2 and P_3 behave such that $E_n-.1 \leqslant \frac{n_2}{N_2} \approx \frac{n_3}{N_3} < E_n$ thereby leading to entries of 25 (average cluster size) and 2 (number of items). Three factors about the columns of Figure V-11c indicate the search results in this example are biased by cluster size:

a) a large range of deviations,

b) a wide distribution of entries throughout this range, and

c) a trend of decreasing cluster size in the column entries.

In this instance bias can be defined by saying that the search procedure, correlation function, and profile properties are such that larger clusters are more likely to rank 1, 2,..., I regardless of their relevancy.

Each of the three considerations listed above is an important factor in determining the presence of search bias. Figure V-11d is an illustration of data lacking some of these characteristics. Column A, having a small range of deviations is an example of unbiased profile behavior. Column B shows a wide range of performance deviations, but no bias because the large range is due to the peculiar performance of one profile. In other words, there is a narrow distribution of entries within the range. Finally, column C shows both a large range of performance ratios and a wide distribution, but lacks a trend in the column of property values. In this case, bias probably exists, but it is not related to the property under consideration.

B. Investigations of Biased Searches

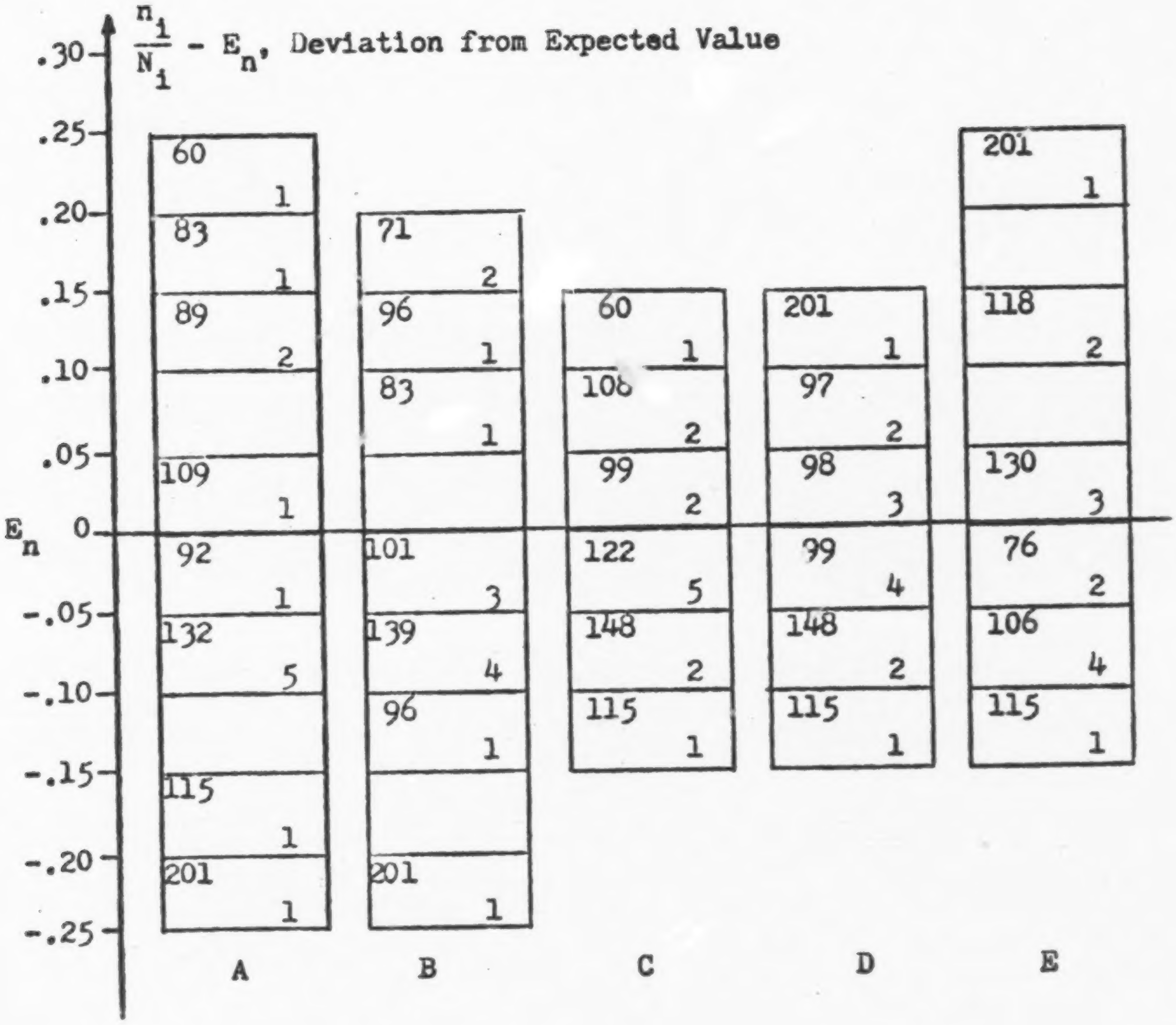
The above procedure is used to analyze searches made with the standard, rank value, and modified profiles described earlier. On level 1 of Hierarchy 1, there is an average of I=4 relevant clusters, so rank positions 1, 2, 3, and 4 are of primary interest. The corresponding figure for level 2 is I=6. In all instances cluster size is the property investigated, although other properties might show the same results since large clusters generally have profiles with many terms, large magnitudes, etc. Cluster size is used because it is invariant among the types of vectors examined. Figures V-12 to V-15 show the behavioral characteristics of rank value profiles in Hierarchy 1. The vectors are the same as those in Section V.3.A, having rank value term weights based on document frequencies. The following observations can be made:

a) unweighted profiles (infinite base value) show a definite bias in favor of small clusters;

b) decreasing the base value reduces the bias significantly; and

c) for the lowest base values, there may be a slight bias in favor of large clusters.

Of great importance is the fact that reduced bias with lower base values is accompanied by the RC-PF performance improvement seen in Figures V-4 and V-5. It is also possible to explain why a base value of 86 is superior to a base value of 66 in the case of level 1 profiles. Figure V-12, shows that the largest cluster (size 201) appears in the initial rank positions as a non-relevant item much more often in the test run with a base value of 66. Hence the expanded profiles (1, 2,..., I) frequently include this cluster, so that all relevant items retrieved must be sifted



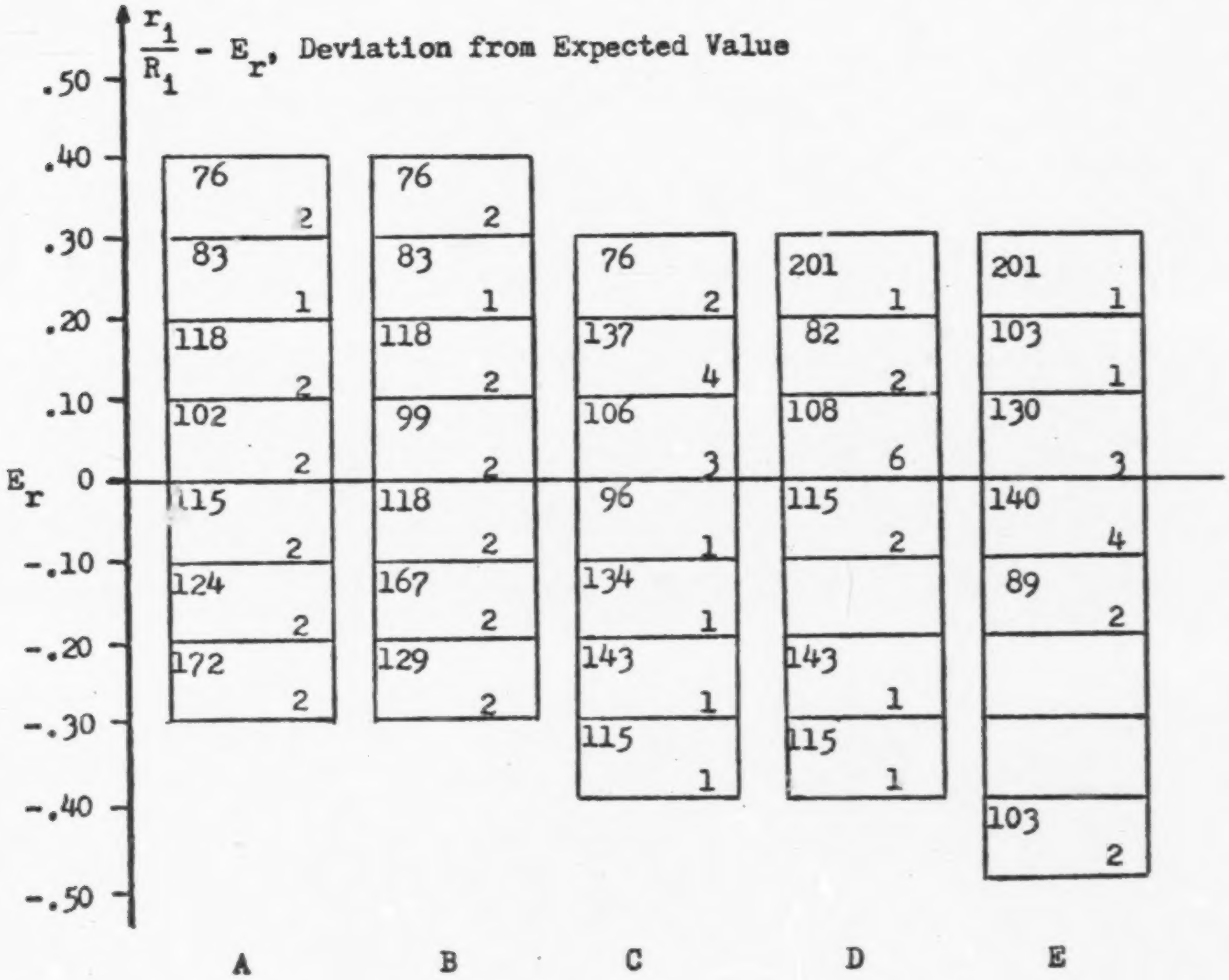
Column	Base Value	E_n
A	∞ (unweighted)	.243
B	226	.228
C	100	.214
D	86	.206
E	66	.208

Behavioral Characteristics of Non-Relevant Profiles

Rank Value Weights Based on Document Frequencies (P_2)

Hierarchy 1, Level 1

Figure V-12



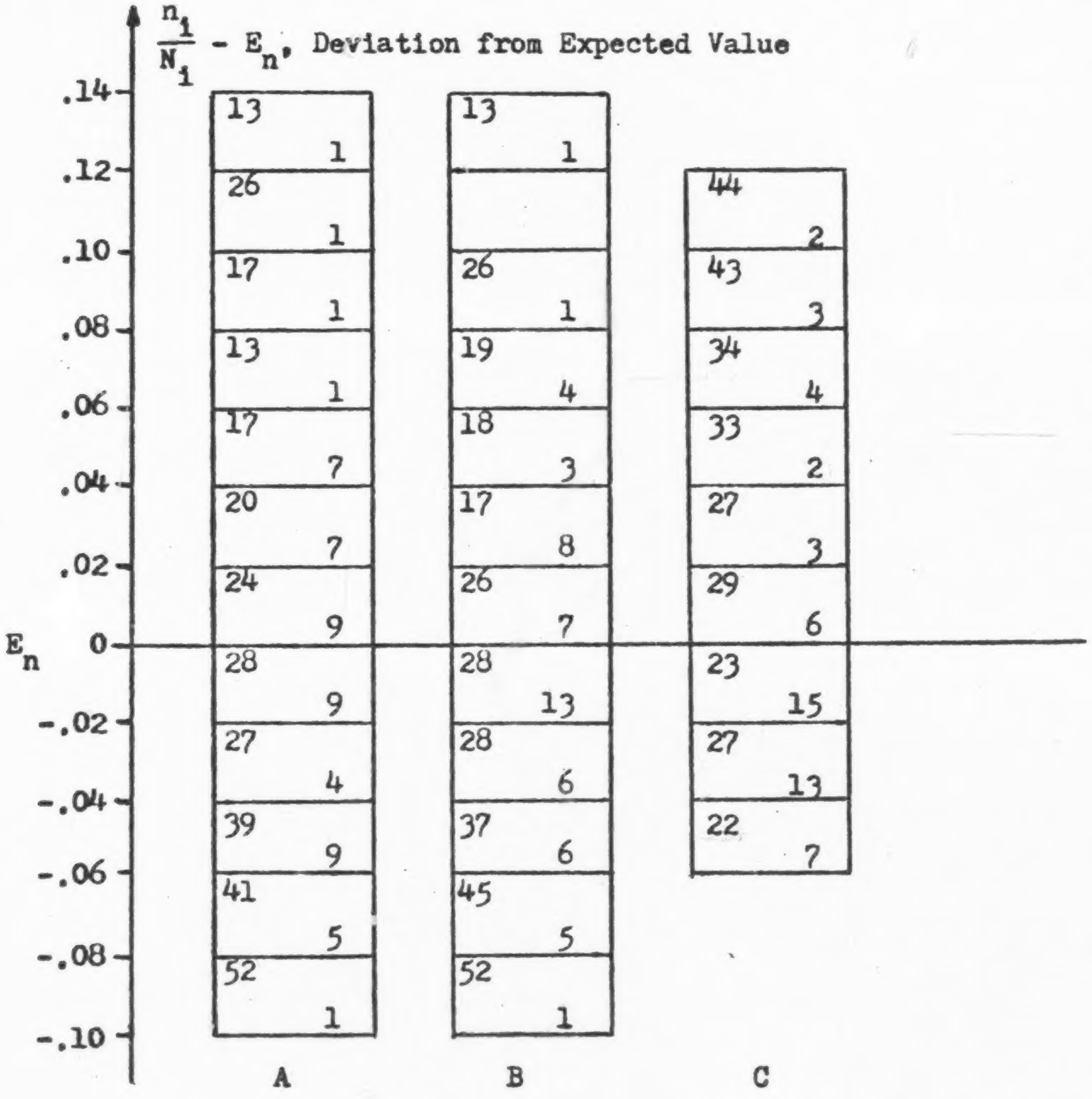
Column	Base Value	E_r
A	∞ (unweighted)	.459
B	226	.494
C	100	.528
D	86	.545
E	66	.540

Behavioral Characteristics of Relevant Profiles

Rank Value Weights Based on Document Frequencies (P_2)

Hierarchy 1, Level 1

Figure V-13



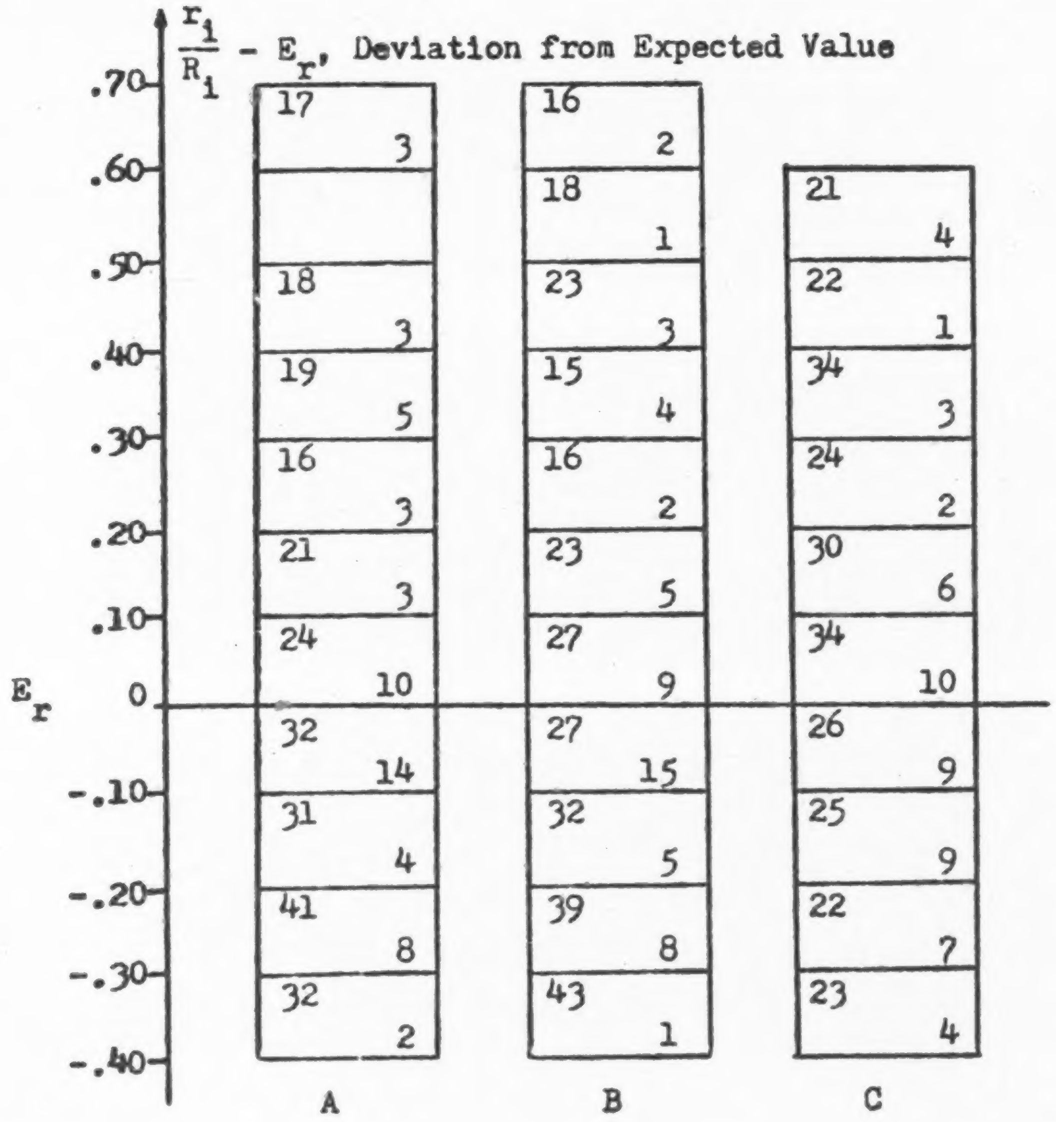
Column	Base Value	E_n
A	∞ (unweighted)	.086
B	100	.084
C	26	.078

Behavioral Characteristics of Non-Relevant Profiles

Rank Value Weights Based on Document Frequencies (P_2)

Hierarchy 1, Level 2

Figure V-14



Column	Base Value	E_r
A	∞ (unweighted)	.320
B	100	.342
C	26	.394

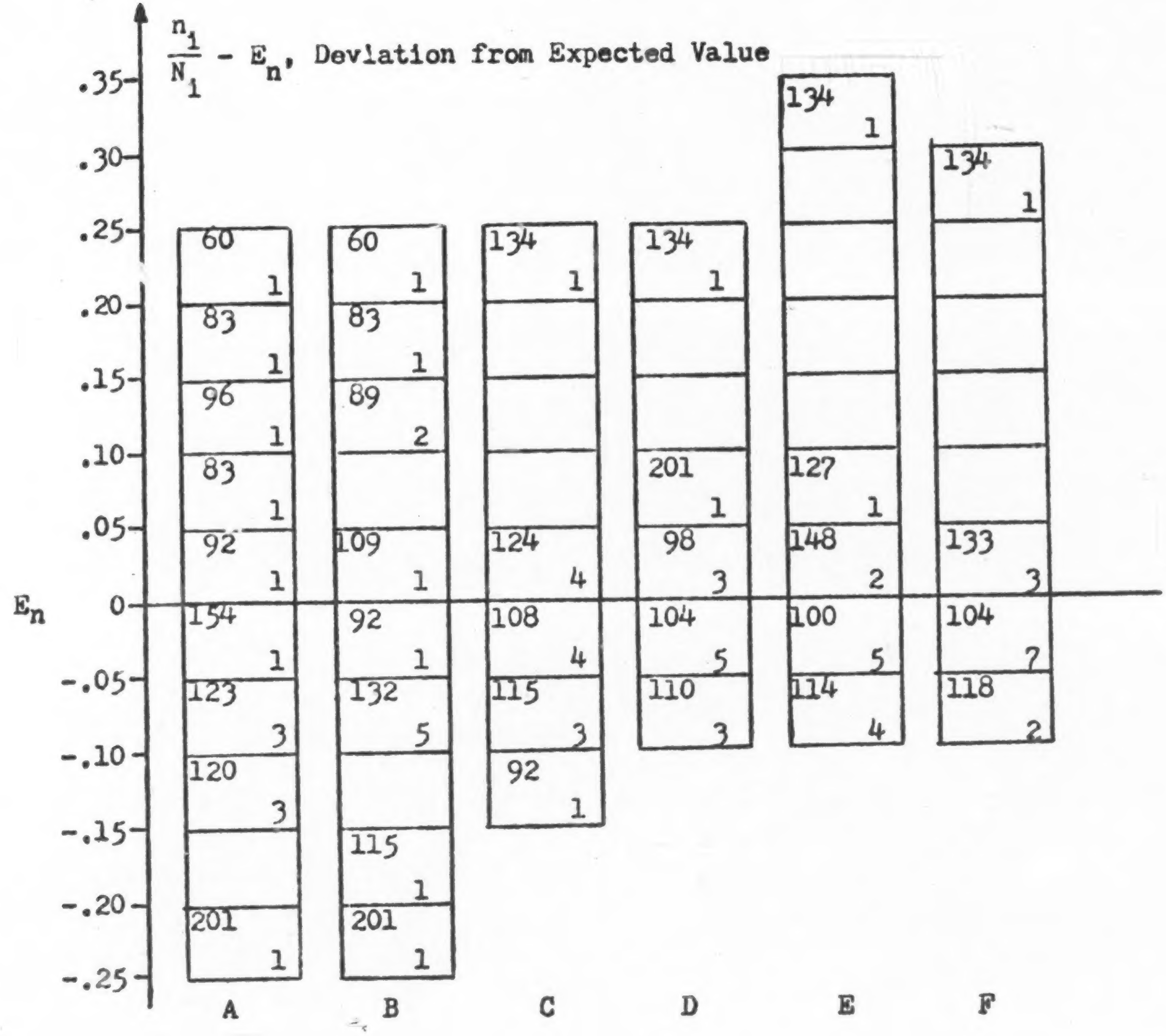
Behavioral Characteristics of Relevant Profiles
Rank Value Weights Based on Document Frequencies (P_2)
Hierarchy 1, Level 2

Figure V-15

out of a large set of other documents. As a result, the values of precision floor measured are significantly lower even though the recall ceiling values might be the same in both cases. This is the exact effect noted in Figure V-4.

To complete the current study of search bias, Figures V-16 to V-19 contain a similar behavioral analysis for the standard profiles (P_1, P_2, P_3) and their counter parts using rank weighting (P_2^*, P_3^*). The two versions of unweighted profiles, one based on P_3 vectors and the other on P_2 vectors, show a large bias in favor of small clusters. In all other cases, practically no bias is present and it is difficult to draw further general conclusions. If attention is focused on the behavior of non-relevant profiles on a particular level and on the range and distribution of entries in the columns, then it is possible to detect more or less bias when term weights are based on document frequencies. However, there is no consensus among levels. It is interesting to note the peculiar, consistent behavior of one vector (size = 134) in Figure V-16. Unfortunately there is no simple explanation for its behavior such as its profile covering a large number of dissimilar documents in a "loose" cluster. A similar phenomenon occurs for a few items on level 2. Another interesting point is the comparison of behavior of the best rank value profiles and P_2^* and P_3^* vectors. Significantly less bias is noted with the latter pair and correlates with the PF-RC performance improvement in Figures V-6 and V-7.

The study of biased behavior has a direct influence on the SMART system which includes factors such as cluster size and hierarchy level in determining a query-profile "correlation". These experiments indicate

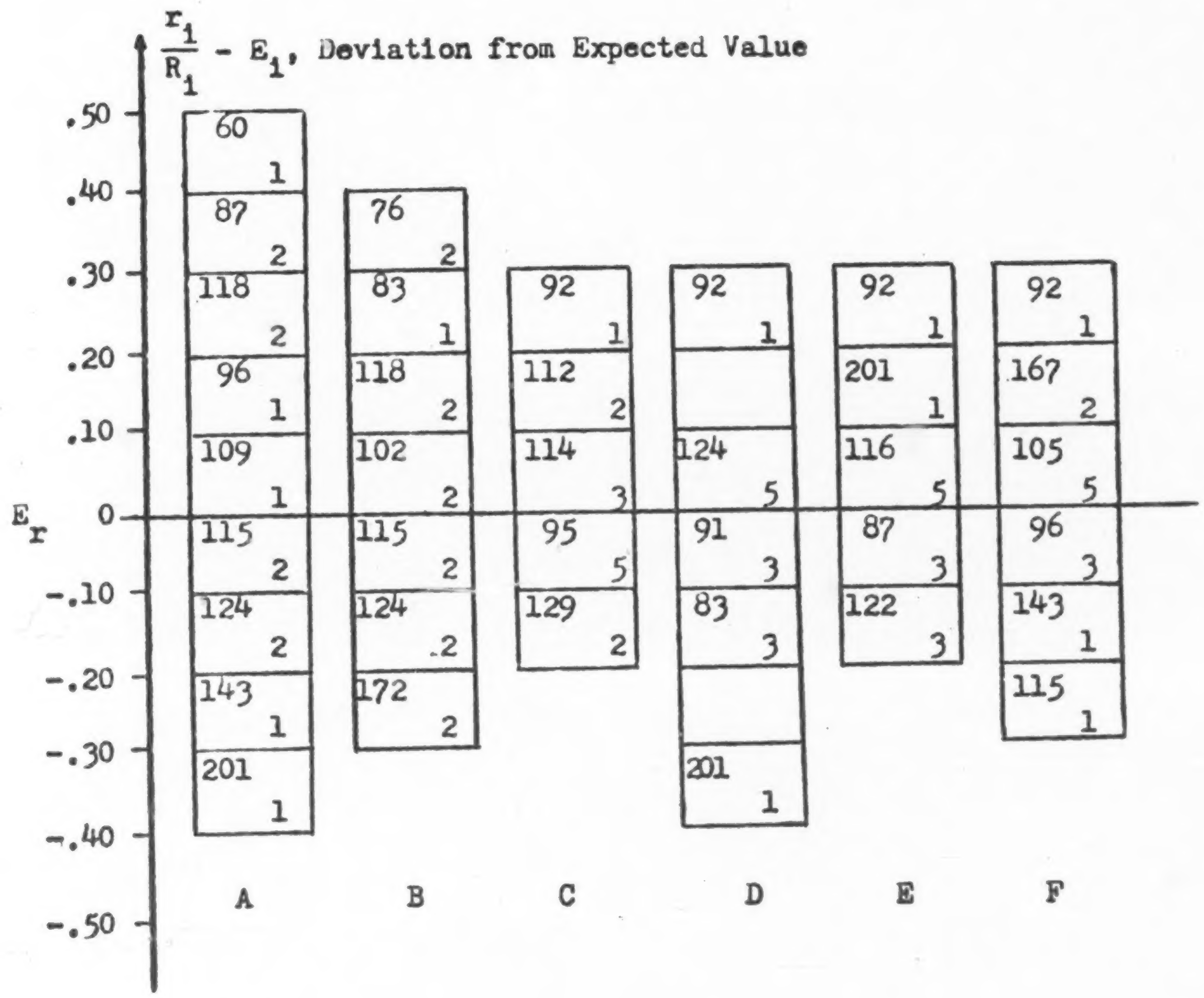


Column	Profile Description	E_n
A	P_1 unweighted (made from P_3)	.246
B	P_1 unweighted (made from P_2)	.243
C	P_3 counts	.213
D	P_2 counts	.212
E	P_3^* ranks	.214
F	P_2^* ranks	.212

Behavioral Characteristics of Non-Relevant Profiles

Standard and Modified Profiles, Hierarchy 1, Level 1

Figure V-16

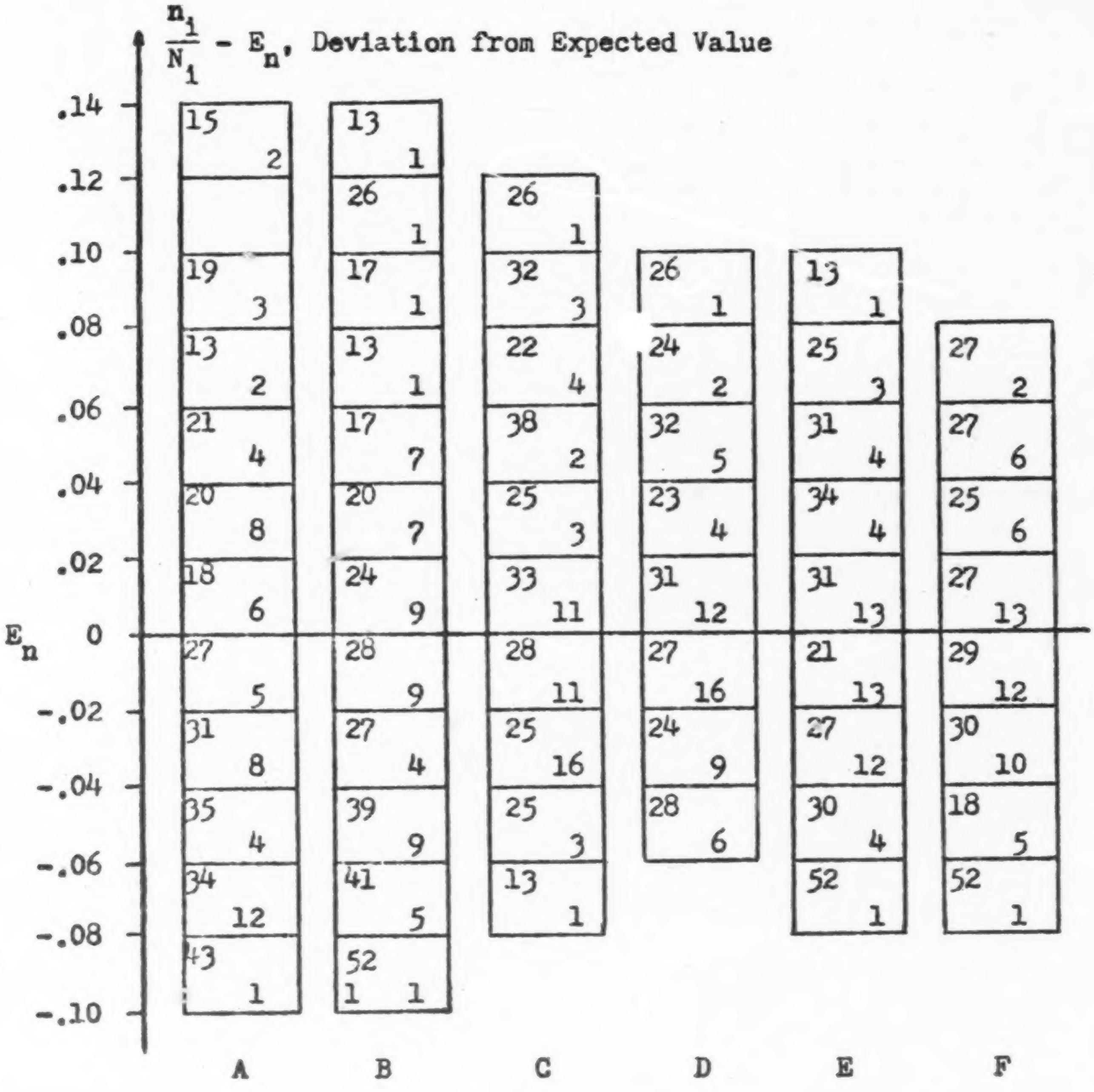


Column	Profile Description	E_r
A	P_1 unweighted (made from P_3)	.453
B	P_1 unweighted (made from P_2)	.459
C	P_3 frequency counts	.529
D	P_2 frequency counts	.530
E	P_3^* frequency ranks	.540
F	P_2^* frequency ranks	.531

Behavioral Characteristics of Relevant Profiles

Standard and Modified Profiles, Hierarchy 1, Level 1

Figure V-17

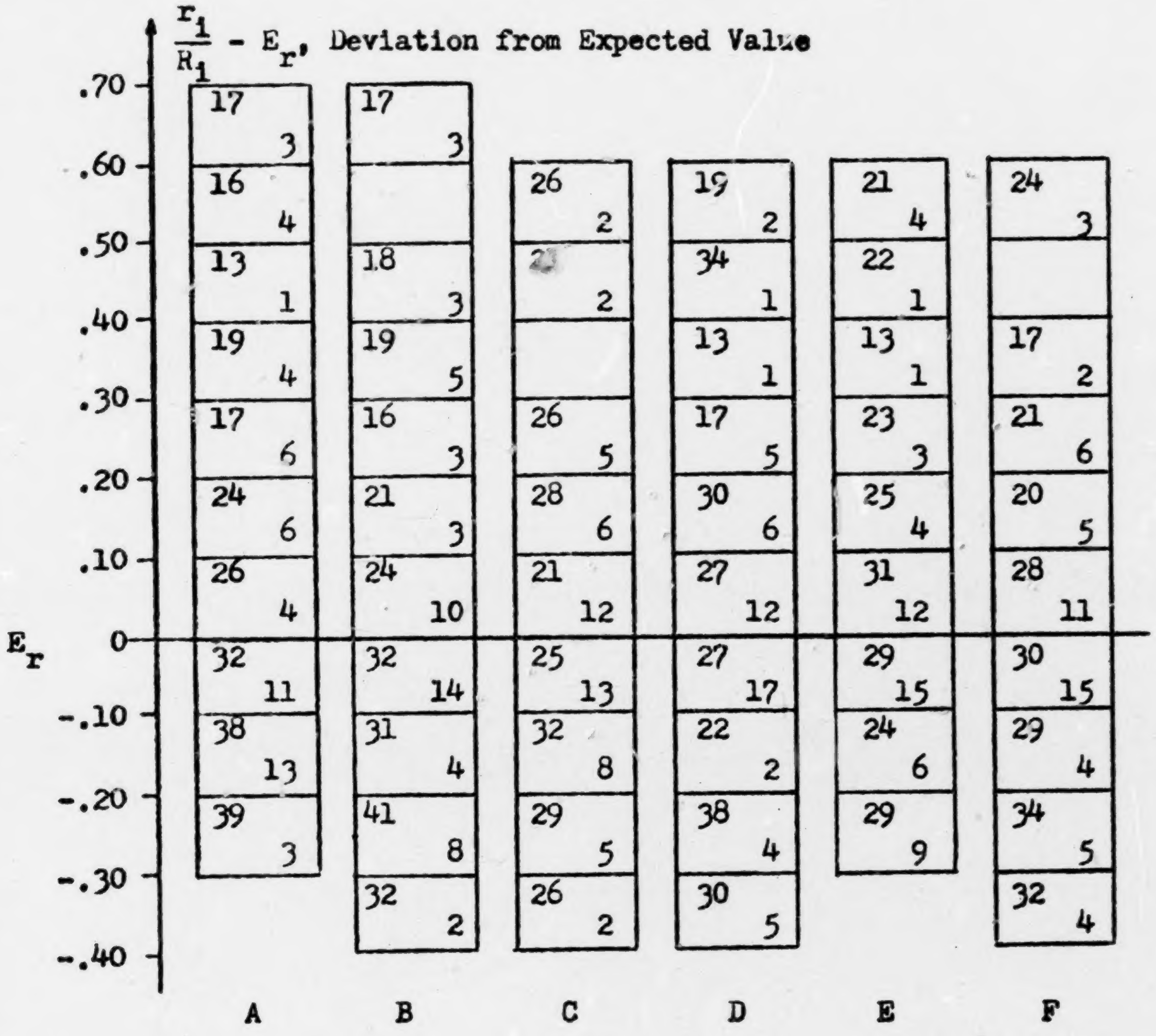


Column	Profile Description	E_n
A	P_1 unweighted (made from P_3)	.089
B	P_1 unweighted (made from P_2)	.086
C	P_3 frequency counts	.079
D	P_2 frequency counts	.080
E	P_3^* frequency ranks	.078
F	P_2^* frequency ranks	.080

Behavioral Characteristics of Non-Relevant Profiles

Standard and Modified Profiles, Hierarchy 1, Level 2

Figure V-18



Column	Profile Description	E_r
A	P_1 unweighted (made from P_3)	.294
B	P_1 unweighted (made from P_2)	.320
C	P_3 frequency counts	.383
D	P_2 frequency counts	.380
E	P_3^* frequency ranks	.404
F	P_2^* frequency ranks	.379

Behavioral Characteristics of Relevant Profiles

Standard and Modified Profiles, Hierarchy 1, Level 2

Figure V-19

that it is possible to derive profiles (P_2, P_3, P_2^*, P_3^*) which give unbiased search performance. Hence, at least for matching items on the same hierarchy level, there is no need to include any factors related to cluster size. Given unbiased profiles, it might be possible to determine the influence of hierarchy level in a similar experiment which mixes the nodes from adjacent levels. In any case, there is a wide variety of properties that could be examined using these techniques.

In summary, this section develops an analysis method for identifying cluster searches which are biased in some manner. The method determines the expected participation of each profile in establishing the search result; bias shows up as a pattern of large behavioral deviations which are related to a specified profile property. Applying this technique to the vectors used earlier provides the following information concerning bias due to cluster size.

a) Unweighted profiles and rank value profiles made with a large base value show a strong bias in favor of small clusters.

b) Reducing the base value or using a fixed weight origin, reduces bias considerably.

c) Within the same hierarchy level, P_2, P_3, P_2^*, and P_3^* profiles show unbiased performance and therefore require no artifical corrections to the cosine matching function as used in the SMART system.

The tests also substantiate the intuitive notions of bias in the previous section and help to explain the results observed there. In all cases thus far, there is a strong correlation between unbiased profiles and good RC-PF performance.

5. Profile Length

By any of the previous definitions, a profile is an aggregate of index terms found in clustered documents. As such, their vectors become longer as cluster size increases. Table V-1 shows that in the first test collection, level 1 profiles average 700-800 terms to characterize 115 documents and level 2 profiles uses 200-300 terms for 28 documents. These figures do not include the initial deletion of terms with unit weights mentioned in Chapter IV. Obviously, considerable disk space is required to store the profiles. In fact, on the IBM 2314 disk unit, level 1 vectors occupy 7 tracks, level 2 vectors need 13 tracks, and all 1500 documents use only 59 tracks. In this case storage overhead for the clustered organization is 34% of the file size and about 35% of this (all vectors on level 1) must be accessed to begin a search. Therefore, in order to make a clustered organization useable in a practical sense, it is imperative to reduce vector lengths in some manner. The following experiments show a large number of terms can be deleted without severely sacrificing search performance.

A particularly simple scheme for reducing vector lengths is to choose a threshold θ and delete all terms with weights less than this value, just as in the case of making the "standard" profiles. This procedure is justifiable since the deleted terms occur only a few times in their respective clusters and certainly did not cause cluster formation. In addition, terms with low weights contribute small amounts to correlations, so that presumably the search results remain relatively unchanged. Furthermore, since the majority of profile terms have low weights anyway, the length reduction is great even if a very low threshold is used. Figure V-20 makes this fact clear by showing the distribution of term frequencies for the same profiles depicted in Figure V-9. In combination, the figures show that at least 71% of the profile terms have individual correlation contributions of less than 0.03 regardless of the weighting scheme used.

If the above deletion scheme is adopted, a reduced form of profile $P = (p_1, p_2, \ldots, p_v)$ can be represented by

$$P' = P - A$$

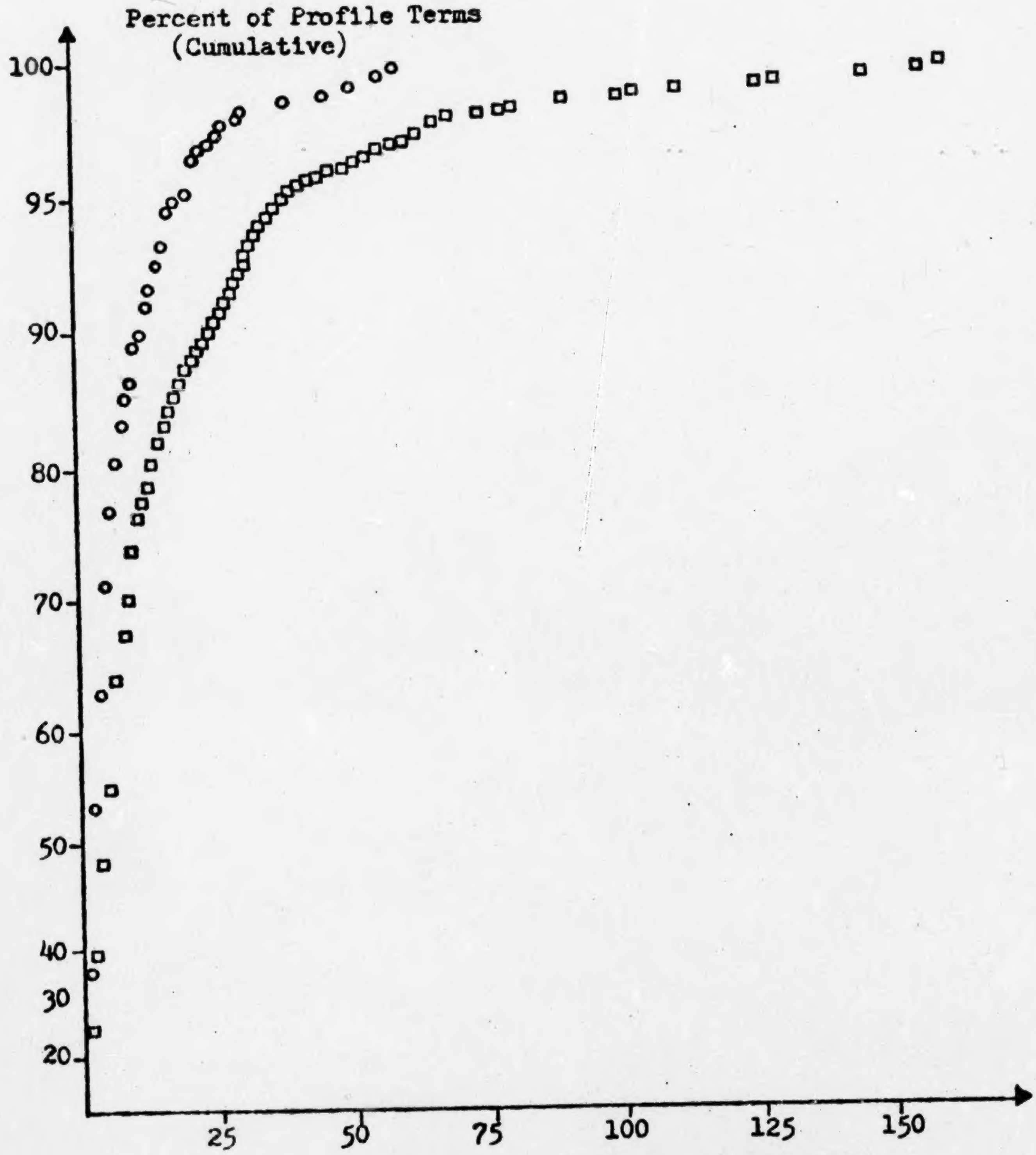
$$A = (a_1, a_2, \ldots, a_v) \quad \text{where} \quad a_i = \begin{cases} p_i \text{ if } p_i < \theta \\ 0 \text{ if } p_i \geq \theta \end{cases} \qquad \text{(V-6)}$$

The vector -A is called the deletion vector. Substituting its value into equation V-2 gives an approximation to correlations using the reduced profile:

$$\text{COS}(Q,P') \approx \alpha \, \text{COS}(Q,P) - \beta \, (\text{COS}(Q,A) \qquad \text{(V-7)}$$

$$\text{where } \alpha = \frac{|P|}{|P + A|} \qquad \beta = \frac{|A|}{|P + A|}$$

When $Q = A$, the maximum correlation loss $|A|/|P|$ occurs. However, queries have characteristics quite different from profiles and it is more meaningful to consider the correlation loss due to specific deleted index terms. Section V.3.C. (equation V-3, in particular) establishes that the correlation contribution of a term (weight p_i) is proportional to $p_i/|P|$. Since θ is the minimum weight retained, the maximum loss per term is bounded



□ Level 1, Profile 5 , Weights = Frequency counts
○ Level 2, Profile 14, Weights = Frequency ranks

Distribution of Profile Terms by Frequency

Figure V-20

by $\theta/|P|$. (Let $\theta/|P|$ be known as the loss ratio whereas values of $p_i/|P|$ are contribution ratios.) This suggests several strategies for obtaining an appropriate cutoff for each profile. For example, a "constant" strategy might establish a maximum tolerated loss ratio L and delete terms such that $(p_i/|P|) < L$. In other cases, the tolerated loss ratio might depend on the distribution of contribution ratios (illustrated in Figure V-9). Specifically considered in Table V-2 are cases in which θ is a function of the mean μ and standard deviation σ of all unique values of $p_i/|P|$ in a given profile.

With the test collections at hand, it is difficult to prove the superiority of one of these strategies. In all cases θ varies among profiles and in the last two instances, perhaps, its value is more sensitive to the characteristics of individual vectors. The constant strategy guarantees that correlations do not change too much for each deleted term; the others simply assure that a reasonable amount of "correlation power" is left in the vector. In spite of all this, the basic question is not one of strategy, but one of showing that a considerable number of terms can be deleted without damaging search performance. In this investigation, the third strategy is used and the parameter δ is changed in the tests on P_3^* profiles for both levels of Hierarchy 1. To review, these profiles have term weights based on term frequency ranks and provide the best performance, thus far. Applying the term deletion strategy to the profiles considerably reduces their length as shown in Table V-3. Their corresponding PF-RC performance curves are shown in Figures V-21 and V-22. The results definitely indicate that a large portion of low weight profile

Deletion Strategy		Tolerated Loss Ratio	Deletion Cutoff
1.	Constant	L	$\theta = L\|P\|$
2.	Fraction of Mean	$\delta\mu$	$\theta = \delta\mu\|P\|$
3.	Deviation from Mean	$\mu + \delta\sigma$	$\theta = (\mu+\delta\sigma)\|P\|$

Note: $|P|, \mu, \sigma$ are obtained for each profile separately;
L and δ are parameters chosen for profile generation

Profile Term Deletion Strategies

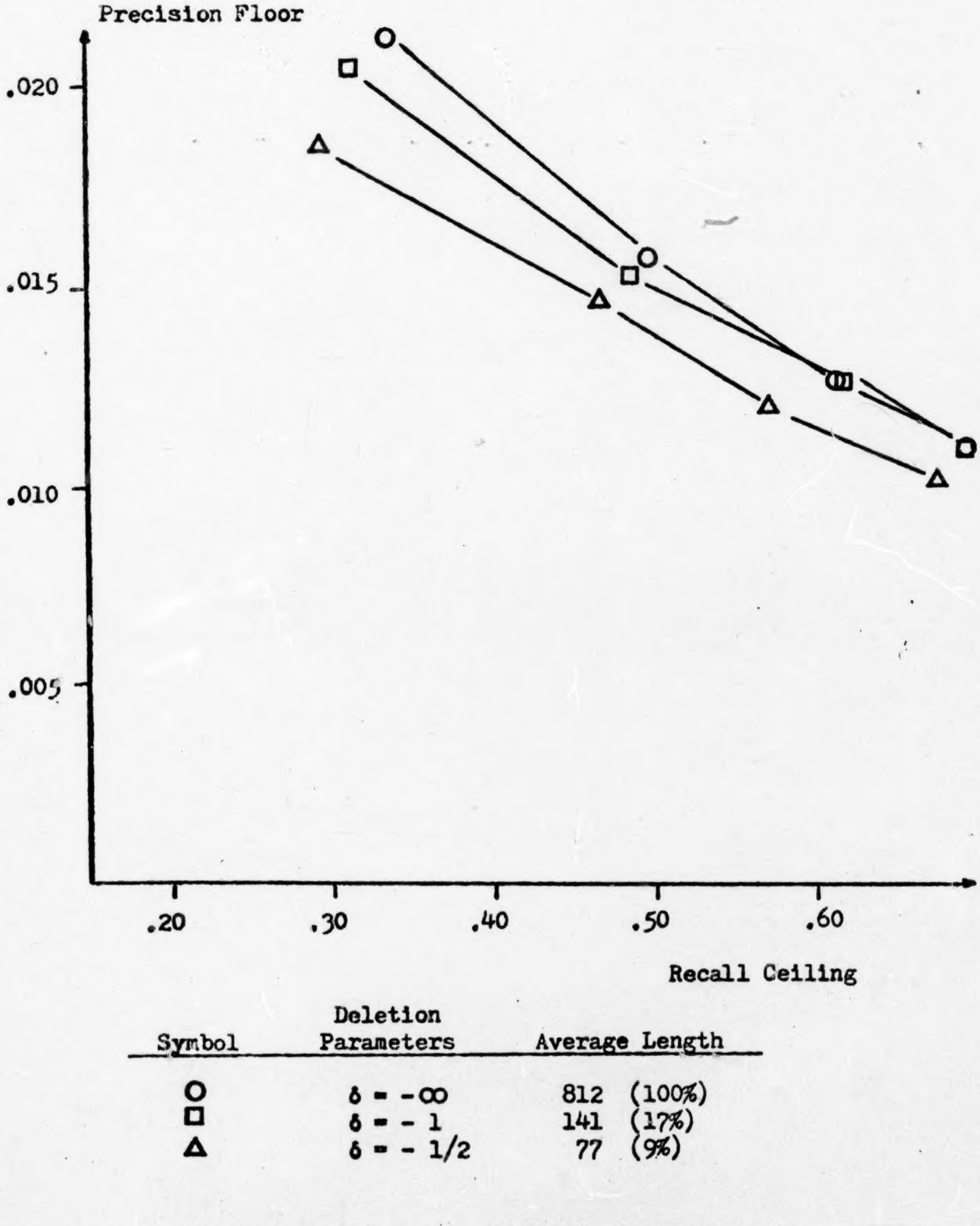
Table V-2

Profile Property	Deletion Cutoff Parameter $\delta = -\infty$	$\delta = -3/2$	$\delta = -1$	$\delta = -1/2$
Level 1, 13 profiles				
Average length	812		141	77
Range of lengths	438-1302		75-200	48-107
Percent of original	100		7	9
Level 2, 55 profile				
Average length	323	171	70	37
Range of lengths	120-672	77-355	33-137	16-76
Percent of original	100	53	22	11

Profile Length Reduction Resulting from Term

Deletion Strategy 3, Hierarchy 1, P^*_3 Profiles.

Table V-3

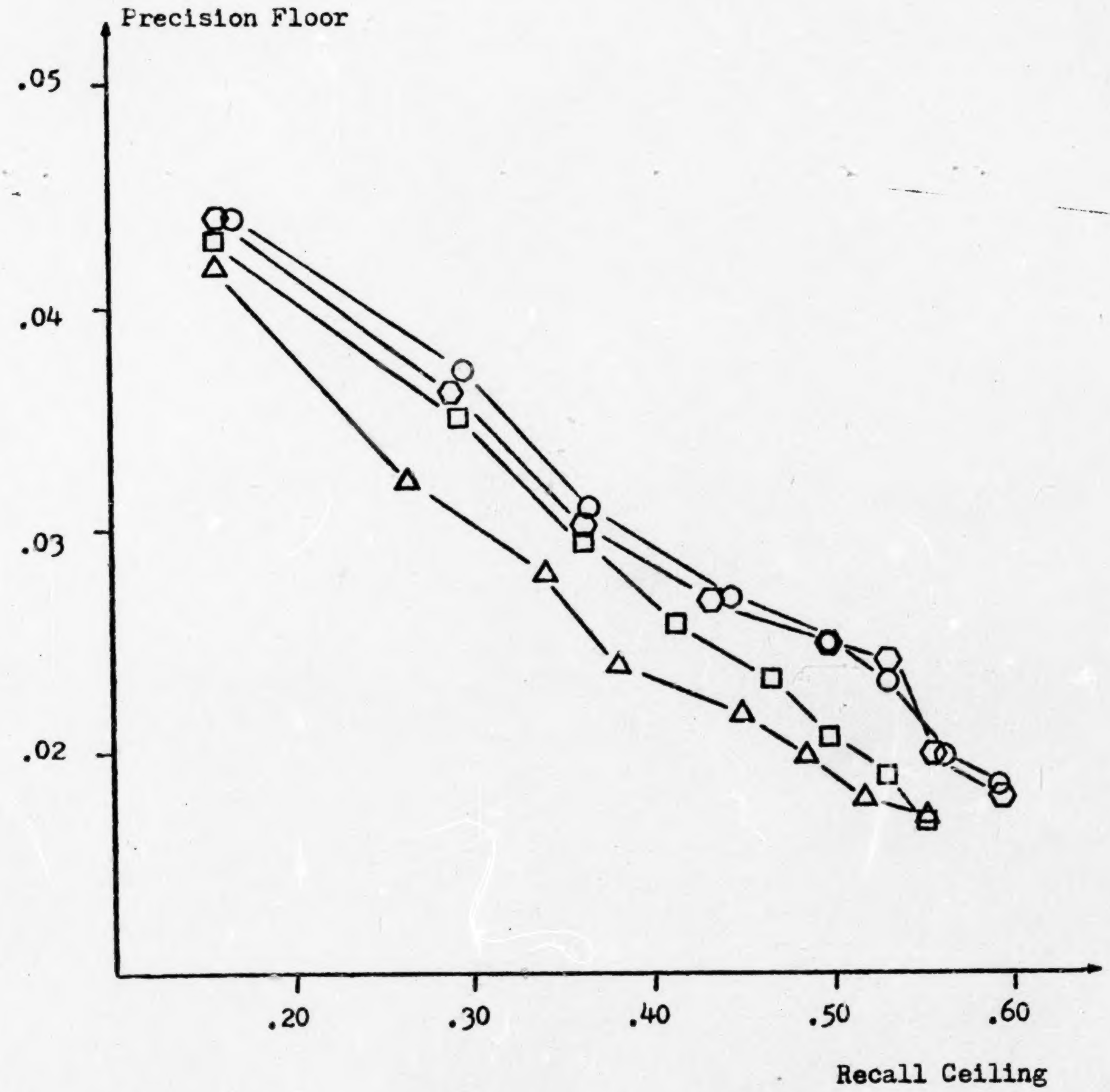


Symbol	Deletion Parameters	Average Length
○	$\delta = -\infty$	812 (100%)
□	$\delta = -1$	141 (17%)
△	$\delta = -1/2$	77 (9%)

Search Performance After Deletion of Profile Terms

P_3^* Profiles, Hierarchy 1, Level 1

Figure V-21



Symbol	Deletion Parameters	Average Length
○	$\delta = -\infty$	323 (100%)
⬡	$\delta = -3/2$	171 (53%)
□	$\delta = -1$	70 (22%)
△	$\delta = -1/2$	37 (11%)

Search Performance After Deletion of Profile Terms

P_3^* Profiles, Hierarchy 1, Level 2

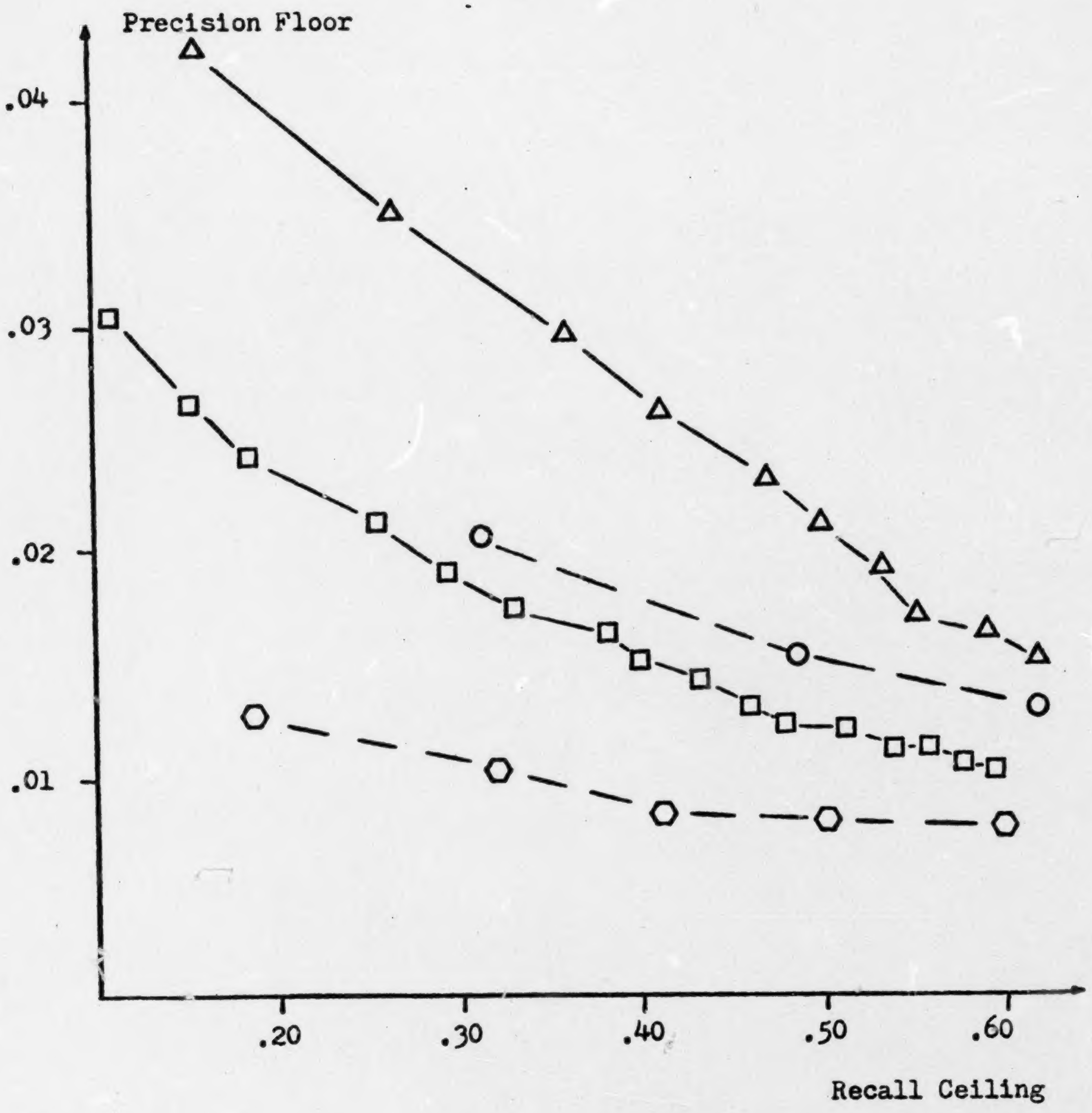
Figure V-22

terms can be deleted with only a small change in performance. This confirms the earlier assertion that the initial term deletion to make "standard" profiles does not alter performance substantially. In some ways, the ability to make extensive deletions is a pleasing result since it implies that the storage overhead and search time can be reduced considerably. For example, with a deletion parameter of $\delta = -1$, level 1 profiles occupy 2 disk tracks and level 2 profiles occupy 4 tracks, making a storage overhead of 7%. Also, using $\delta = -1$ results in a performance drop of about 1% - 3%, somewhat less than the improvement found between P_3^* and P_3 profiles. In other ways, it is unsettling that so many terms can be deleted with so little effect. If it were known that the best possible profiles were being used, then large deletions would not be bothersome. On the other hand, all of the profiles considered need a great deal of improvement in order to reach the best achievable performance (Figures IV-8 and IV-9) and if such improvements were made, large deletions might yield disastrous search results.

In Litofsky's experiments (2), a profile includes only terms which are common to all documents in its crown. During clustering, unweighted vectors (P_1) are used; however once the hierarchy is made, profiles are reformatted starting at the lowest level and proceeding upward. In the reformatting, all profiles with the same parent (filial profiles) have their common terms removed and retained only in the parent profile. Applying this procedure to all vectors removes many terms and decreases the storage overhead for the file. To accommodate the altered structure, the search strategy is changed so that it follows all paths which match the query in any manner. This is an appropriate technique since each profile term applies to all information beneath it.

Litofsky presents no evaluation data related to his profiles, but it is possible to approximate these vectors and to evaluate them using the SMART procedures. Specifically, the highest frequency terms in each SMART profile are eliminated in order to simulate their transfer to parent profiles. Using the algorithm developed earlier, the deletion cutoff for each profile is determined from its distribution of term correlation ratios. A cutoff $\theta = (\mu + \gamma\sigma)|P|$ is applied and all terms with weights larger than θ are removed. This procedure guarantees that the deleted terms are common to most members of a filial profile set and therefore are the terms likely to be removed in Litofsky's original scheme. Obviously the deleted terms are different in each vector. One additional difference, of course, is the use of weighted profiles in these experiments. In this regard, the cutoff limits the affected terms to those with about the same degree of significance in all subordinate information. This is an important factor in interpreting the evaluation curves in Figure V-23. In all cases, the test vectors are of the $P^*_3(\delta = -1)$ variety; the high weight deletion parameter is set at $\gamma = 1$. On level 1, the additional deletion removes an average of 15 terms, each occurring in 3 of the 13 profiles. On level 2, an average of 6 terms are removed, each occurring in 1.7 of the 4.2 sons in a filial set.

The large performance loss observed in Figure V-23 leads to the following observations. First, major profile terms cannot be removed from some parts of the hierarchy and retained in others, especially within



Symbol	Level	Deletion Parameters	Average Length
○	1	δ = -1, γ = ∞	141 (100%)
⬡	1	δ = -1, γ = 1	126 88%
△	2	δ = -1, γ = ∞	70 100%
□	2	δ = -1, γ = 1	64 91%

Performance Loss Due to Deletion of High Weighted

Profile Terms, Hierarchy 1, P_3^* Profiles

Figure V-23

the same filial set. All parts of the hierarchy must be treated alike as shown in the following example. Consider two profiles P and P', of which the first contains term T and the second has T elevated to its parent node. Now consider a query containing T which passes through the upper hierarchy via some search path and which is matched with both profiles. The query matches T in P, but not in P' even though the term is probably more characteristic of the latter cluster. In the case of P', the match is recorded on some previous level and this information might be carried along in the search. However, this places a great emphasis on the problem of relating the importance of T in the parent to its importance in individual subordinate profiles. *This leads to the second observation, namely that individual weights of major profile terms are important in differentiating profiles from all other vectors, filial or non-filial*. In practice most profiles contain a great many common terms (e.g., 70%) and their weight distribution is the primary differing characteristic. Under Litofsky's scheme, common term occurrences are coalesced in the parent in a way that obscures the important weight differences in subordinate profiles. The easiest and probably best solution to this problem is to avoid altering the profile in the beginning. Whereas Litofsky's scheme may be viable for simple, unweighted profiles, it is definitely to be avoided in more sophisticated systems.

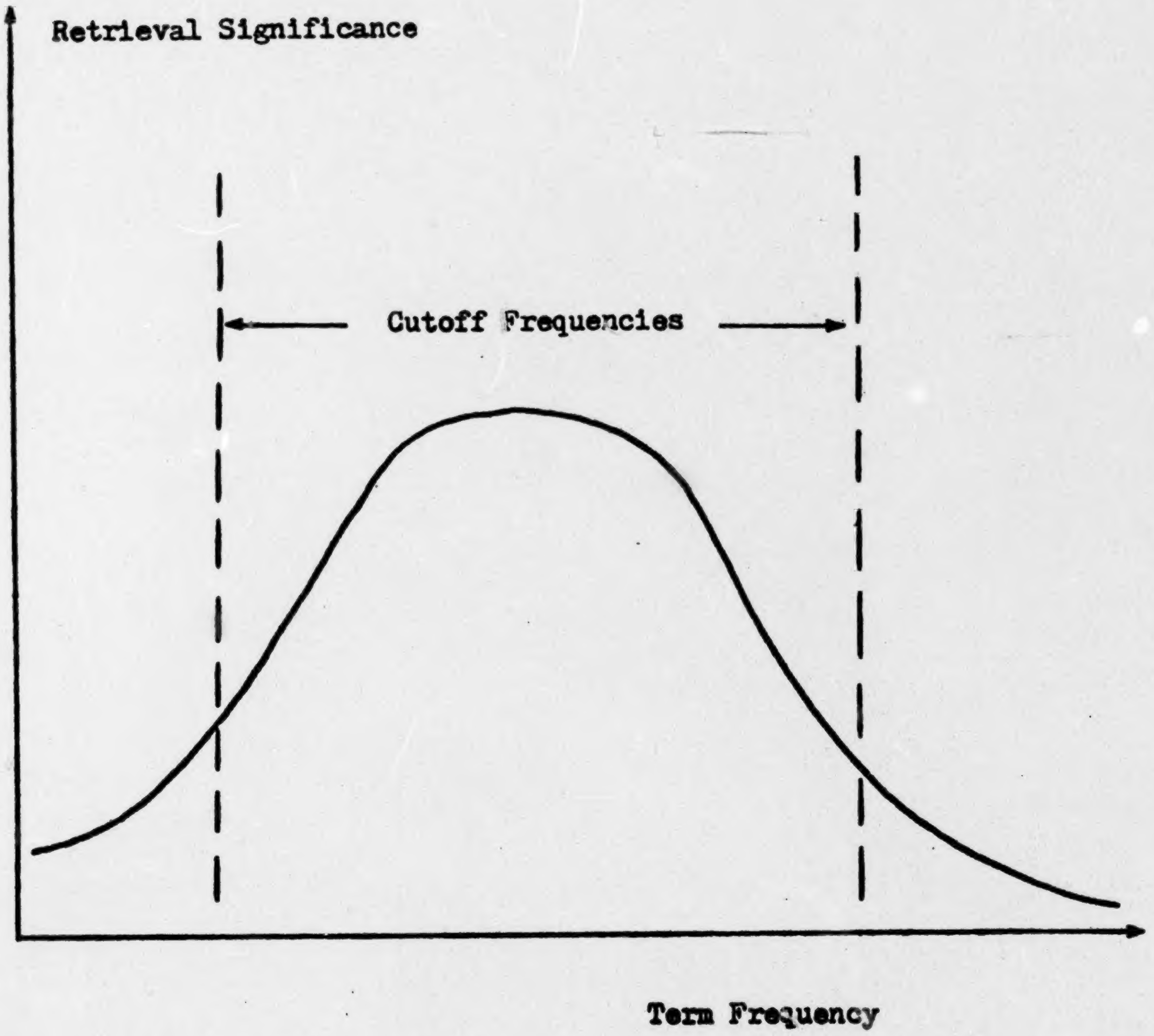
To summarize, *profiles can be subjected to considerable deletion of low weight (less frequent) terms with little change in the quality of search output*. A number of procedures are suggested for deriving a deletion cutoff which guarantees that correlations remain reasonably undisturbed.

Experiments with one method ($\theta = (\mu - \sigma)\cdot|P|$) indicate that the deletion of 80% of the lowest weight terms drops the RC-PF measures only 1% to 3%. On the other hand, an attempt to remove or combine related occurrences of high weight profile terms results in much poorer performance. Such procedures are to be avoided.

6. Frequency Considerations

Up to the present, finding an adequate profile has been handled as a problem of indexing a "super document" composed of the clustered documents. Using this analogy, the P_1, P_2 and P_3 profiles are extensions of conventional indexing techniques. This research suggests modifying the standard definitions a bit in order to achieve better results. In all cases, however, the importance of a term to retrieval--that is, the amount its match contributes to the total cosine correlation--is non-decreasing with frequency. Many retrieval experts dislike using a monotonic relationship between frequency and importance. Consequently, this section considers profiles in which term importance (correlation contribution) is not monotonic, but first increases and then decreases with total frequency. Contrary to expectations, performance decreases under these conditions and the monotonic relationship is established as a better approximation to the true association.

As background, consider the work of H. P. Luhn selecting terms for an indexing vocabulary for a set of documents. Hypothesizing a relationship between frequency and retrieval importance such as that in Figure V-24, Luhn argues that words with high or low frequency have little significance and therefore, can be eliminated (3). Recent experiments by



Luhn's Hypothetical Relationship Between

Retrieval Significance and Term Frequency

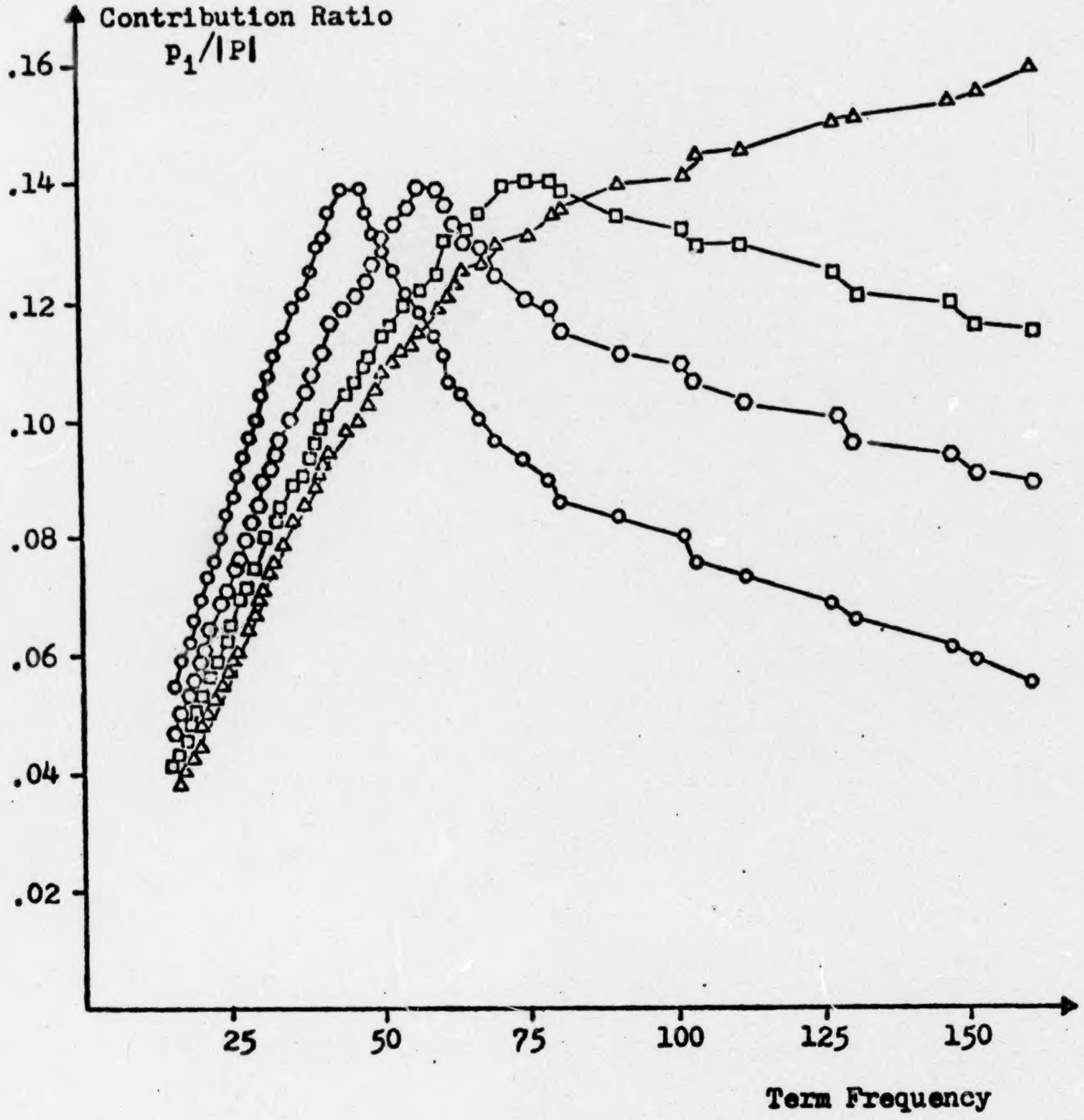
Figure V-24

Bonwit and Aste-Tonsman measure the relationship between retrieval results and the statistical properties of a collection's vocabulary (4). Among other things, they concur with Luhn that generally the most discriminating terms have mid-range frequencies.

If the mid-frequency terms are the best descriminators in the entire collection vocabulary, and therefore are the most important keywords, then a logical progression is to extend this concept to the mini-vocabularies of individual profiles. That is, given the profile index terms and their frequencies, the largest weights should be assigned to terms with the middle frequencies. Consequently, the correlation contribution ratio $p_i/|P|$ from a matching profile term increases and then decreases as the term involved has greater frequency. The distribution of contribution ratios for some of the profiles used in this section are shown in Figures V-25 and V-26. The method for producing the increasing-decreasing behavior is given below; for now it is sufficient just to note the general shape of the curves.

So far, this research has not contradicted Luhn's original concept as extended to individual clusters or the mini-vocabularies in their profiles. Clearly, many low frequency terms can be eliminated without much effect on retrieval. Deletion of high frequency profile terms was tried in the previous section and produced very poor results. However, a small number of very common words (a, the, as,...) are removed prior to experimentation and these are probably the terms Luhn would eliminate on the high end of the frequency spectrum.

The remaining task is to evaluate the effectiveness of profiles whose mid-frequency terms provide the largest correlation contributions.

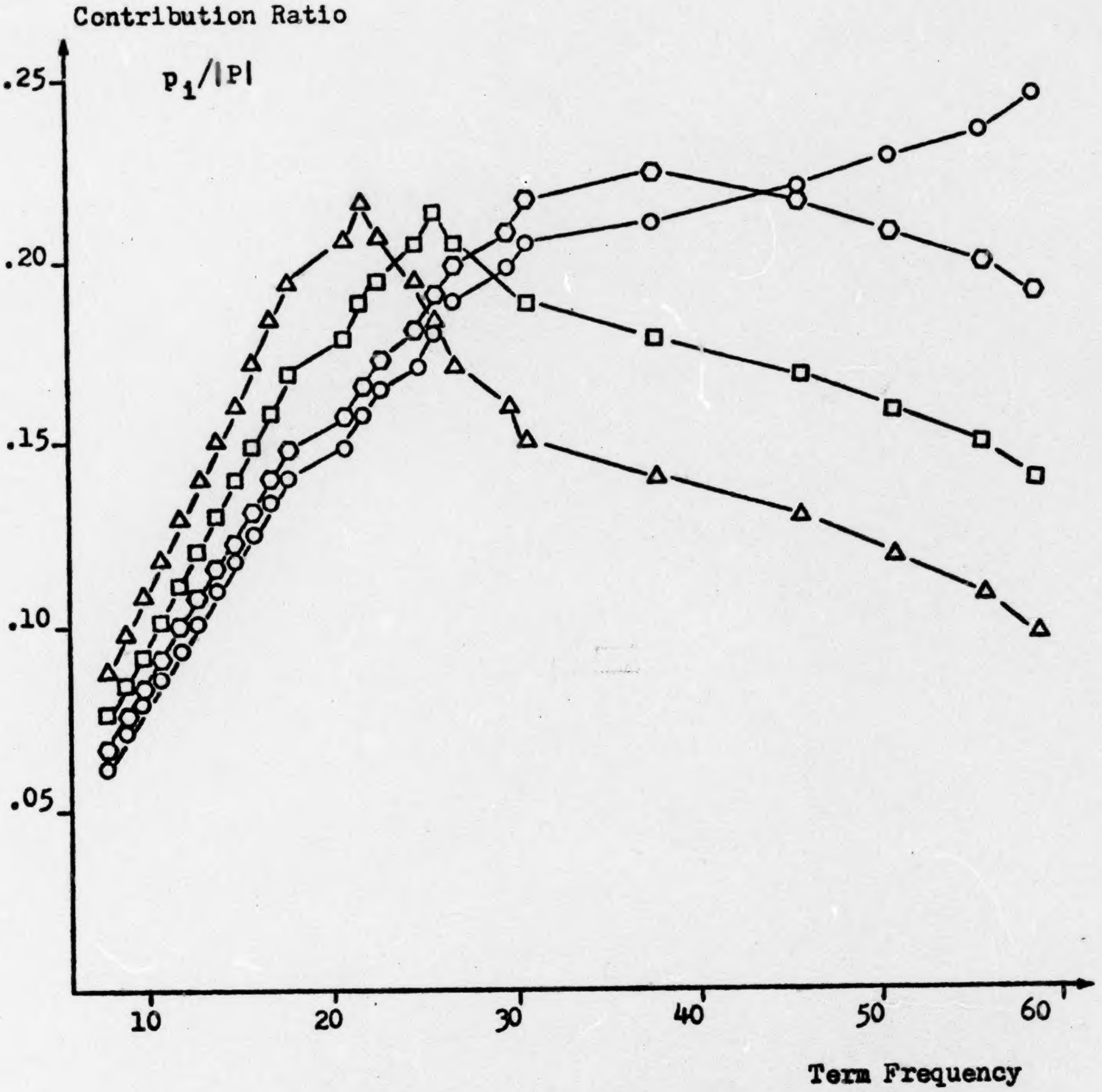


Symbol	Parameter	Max. Weight θ (rank)	Number of Affected Terms
△	$\xi = \infty$ (none)	65	0
□	$\xi = 1$	54	11
○	$\xi = \frac{1}{2}$	47	22
◦	$\xi = 0$	40	35

Contribution Ratios Resulting from Bending

Hierarchy 1, Level 1, Node 5, $P_3^*(\delta = -1)$ Profile, 155 Terms

Figure V-25



Symbol	"Bend" Parameter	Max. Weight θ (rank)	Number of Affected Terms
△	$\xi = \infty$ (none)	31	0
□	$\xi = 1$	26	5
⬡	$\xi = \frac{1}{2}$	22	9
○	$\xi = 0$	20	15

Contribution Ratios Resulting from Bending

Hierarchy 1, Level 2, Node 14, $P_3^*(\delta = -1)$ Profile, 62 Terms

Figure V-26

This condition is approximated in the following experiments by reweighting terms to "bend" the normal monotone contribution curves into an increasing, then decreasing shape. The input vectors are of the ($P_3^*(\delta = -1)$ variety so weights are based on frequency ranks and some term deletion has occurred. Briefly, a maximum allowable weight θ is established for each profile; and any larger term weight $\theta + x$ $(x > 0)$ has its value lowered to $\theta - x$. A specific description of the re-weighting algorithm for an input profile $P = (p_1, p_2, \ldots, p_v)$ is as follows:

a) calculate the mean μ and standard deviation σ of all unique values of contribution ratios $p_i/|P|$;

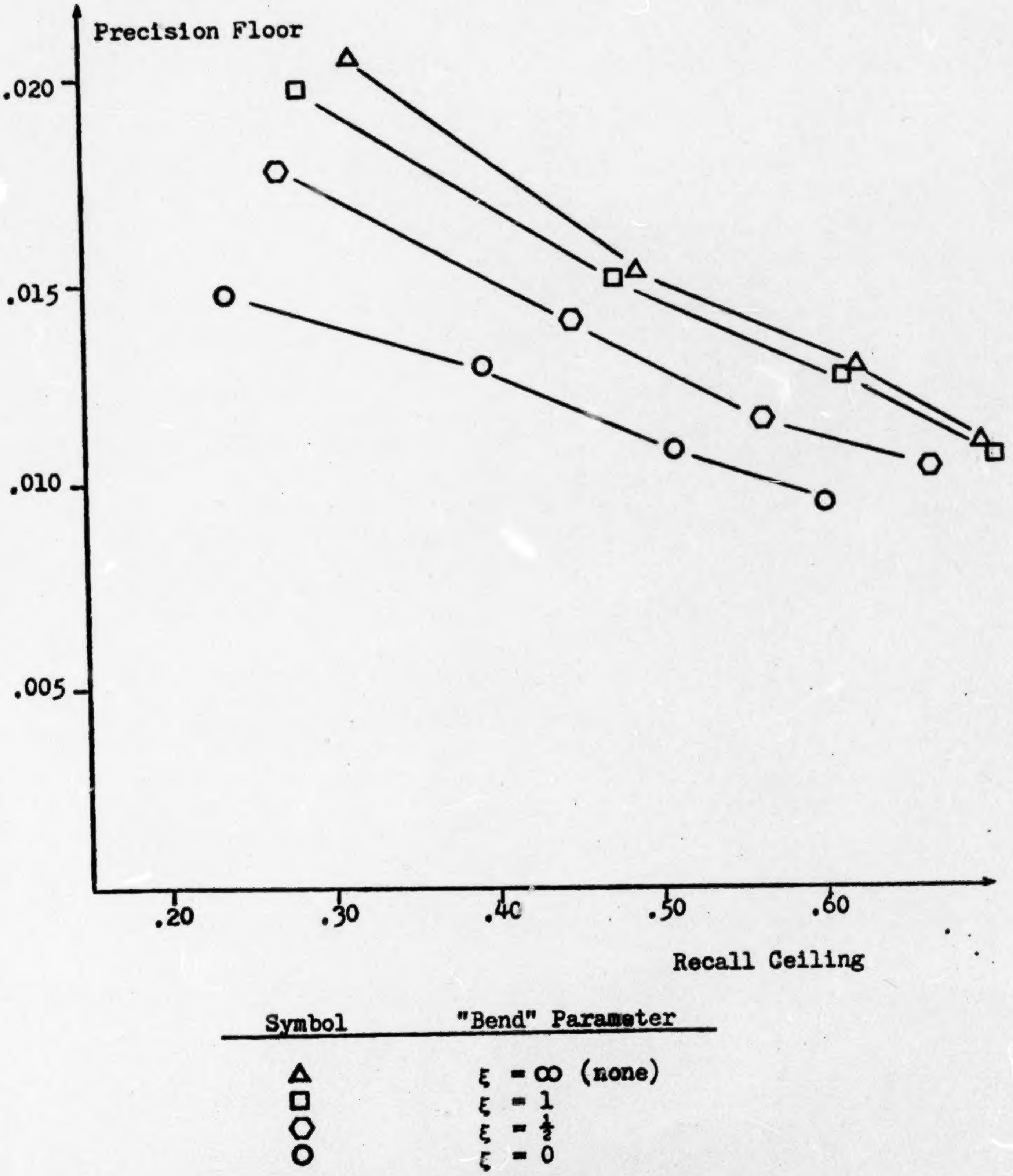
b) determine the maximum allowable weight (or bend point) as $\theta = (\mu + \xi\sigma)\ |P|$ where ξ is a constant parameter chosen in advance;

c) re-assign term weights according to

$$p_i = \begin{cases} p_i \text{ if } p_i \leq \theta \\ 2\,\theta - p_i \quad \text{if} \quad p_i > \theta \end{cases}$$

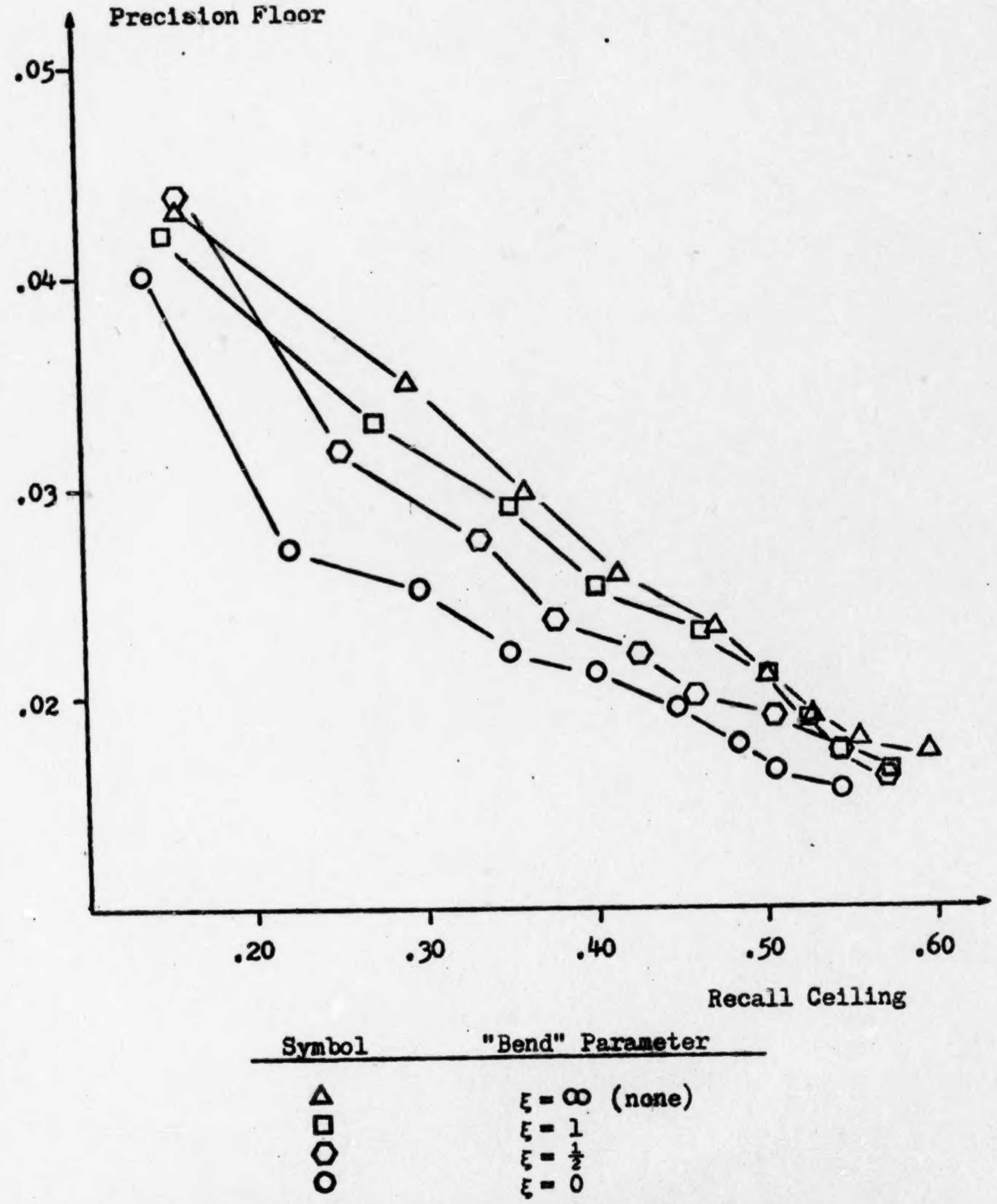
Figures V-25 and V-26 show the original and altered contribution curves for typical profiles ($\xi = \infty$, 1, 0.5, 0). In all cases, the altered profiles assign the most importance (largest contribution ratios) to terms of middle frequencies. The parameter ξ controls the number of affected terms.

Evaluation curves for the altered profiles (Figures V-27 and V-28) suggest that performance drops steadily as the contribution curve receives greater bending (ξ and θ decrease). Thus, it appears that term weights within individual profiles should not decrease with



Search Performance Resulting from Profiles with Increasing-Decreasing Contribution Curves-Hierarchy 1, Level 1, $P^*_3(\delta = -1)$ Profiles

Figure V-27



Search Performance Resulting From Profiles with Increasing-Decreasing Contribution Curves-Hierarchy 1, Level 2, $P_3^*(\delta = -1)$ Profiles

Figure V-28

frequency, but should increase or remain constant. It must be said that part of the performance loss is due to the fact that the characteristics of each profile determine its degree of modification. Thus, a term may have its weight substantially altered in one case and remain unchanged in another. On one hand, earlier tests advise against different treatment of individual profiles. On the other hand, strict adherence to this rule neglects natural variations in term importance due to different cluster contents. So, even though Luhn and Bonwit and Aste-Tonsman indicate that the mid-frequency terms are the best discriminators within a collection vocabulary, this concept does not carry over to the indexing of individual vectors. In the latter situation, high frequency terms have at least as much significance as mid-frequency terms. At the very least, term weights within individual profiles should be non-decreasing with frequency.

7. Unweighted and Partially Weighted Profiles

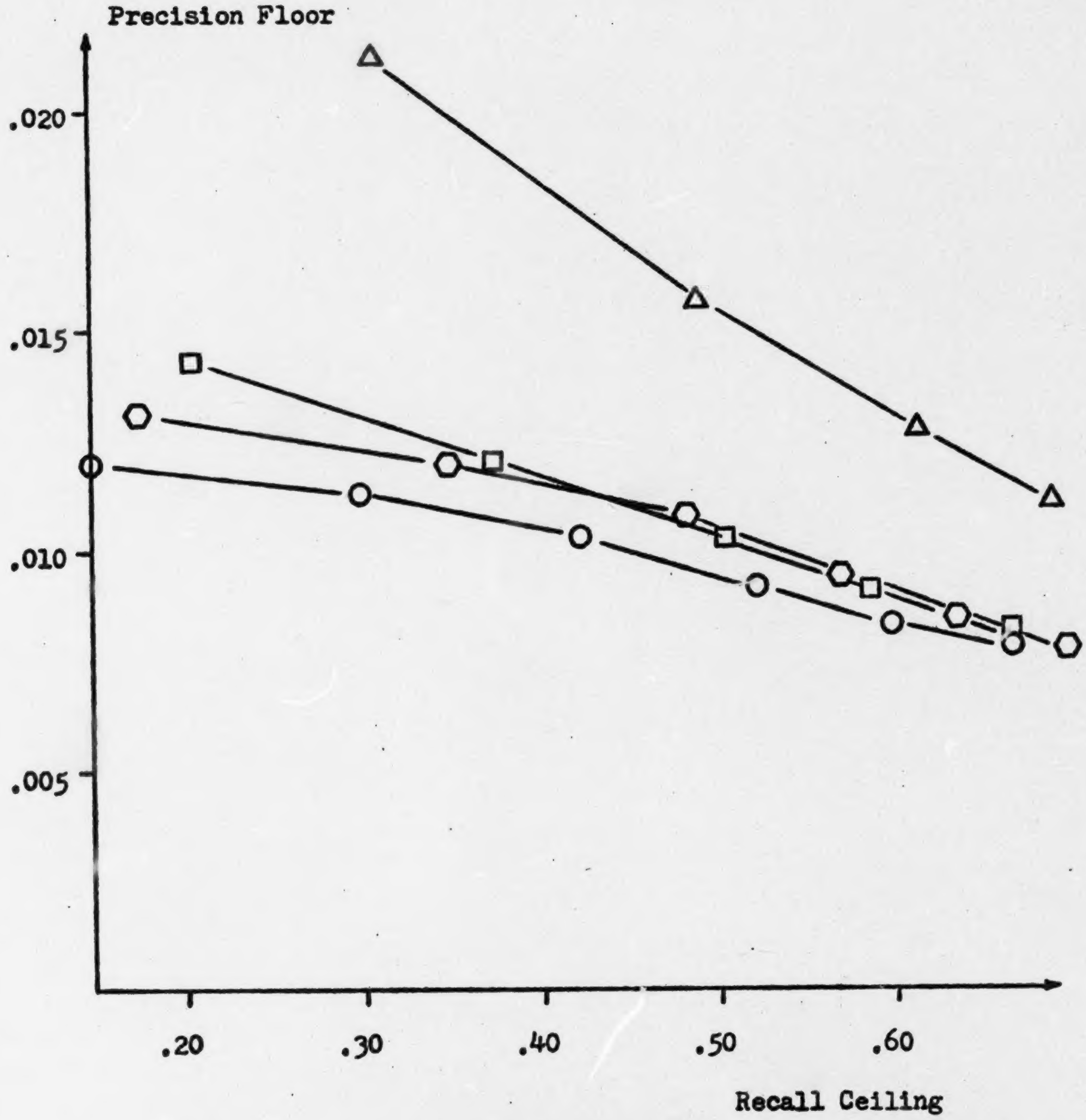
From the start, unweighted profiles (P_1) demonstrated poor search performance and results which were biased in favor of small clusters. In spite of this, unweighted profiles merit additional attention because of their simplicity and storage economy. The storage considerations are not minor, in the SMART system at least, where weights require as much memory as index term identifiers (a S/360 halfword each). This section describes a number of attempts to correct the performance deficiencies of unweighted profiles while retaining their other advantages.

To review, Section V.4 develops a technique for detecting search results which are biased with respect to a specific profile property. Tests made on unweighted profiles show such searches favor the retrieval of small clusters regardless of their relevancy. A straightforward scheme to remove bias is to alter the cosine matching function to give larger correlations when large clusters are involved. Specifically, given a profile P of size S (number of documents in its crown) and a query Q, a modified cosine value is computed from the equation

$$\mathrm{MCOS}(Q,P) = S^{\tau}\left[\mathrm{COS}(Q,P)\right] \qquad \text{(V-8)}$$

where τ is an experimentally determined constant. Figure V-29 shows the effect of using the MCOS function on unweighted profiles for Hierarchy 1, level 1; the P_3^* curve is included for comparison. Figure V-30 shows the data related to biased behavior in the searches. Combining this information yields the following observations. First, the modified cosine does improve performance slightly as τ increases and at the same time reduces bias toward small clusters. It is doubtful, however, that much additional improvement can be made by increasing τ further. Second, the behavior of P_1 profiles with $\tau = .2$ is an example of unbiased, but poor search results. Earlier tests show that good performance is free of bias; obviously the converse does not hold. Third, even though Figure V-30 shows no strong size bias for $\tau > 0$, bias may exist with respect to another profile property. This is a likely situation since the range of deviations from E_n is large and has a broad distribution of entries.

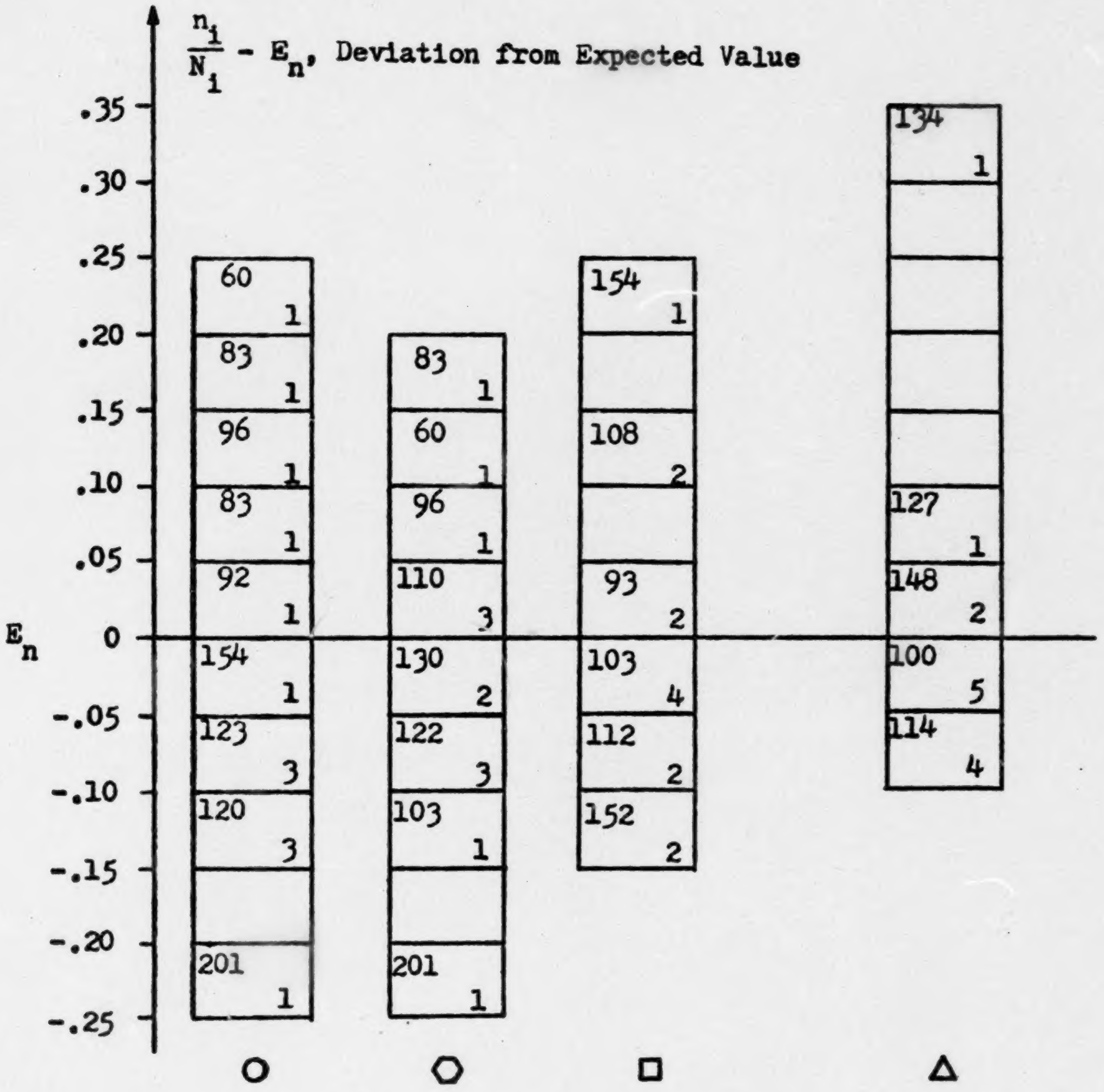
Additional investigation of bias in unweighted profiles led to the following insight and experiment. Because the documents are grouped, some terms in each profile are more characteristic of a cluster than others. In fact, this work has indicated that a large number of terms are completely unnecessary. However, an unweighted vector assigns equal significance to all terms; in a vector of length k all terms have a correlation



Symbol	Profile Description	MCOS Parameter
○	P_1 unweighted	$\tau = 0$
⬡	P_1 unweighted	$\tau = .1$
□	P_1 unweighted	$\tau = .2$
△	P_3^* weighted	

Performance of Unweighted Profiles Using a Size Dependent Cosine Function-Hierarchy 1, Level 1

Figure V-29

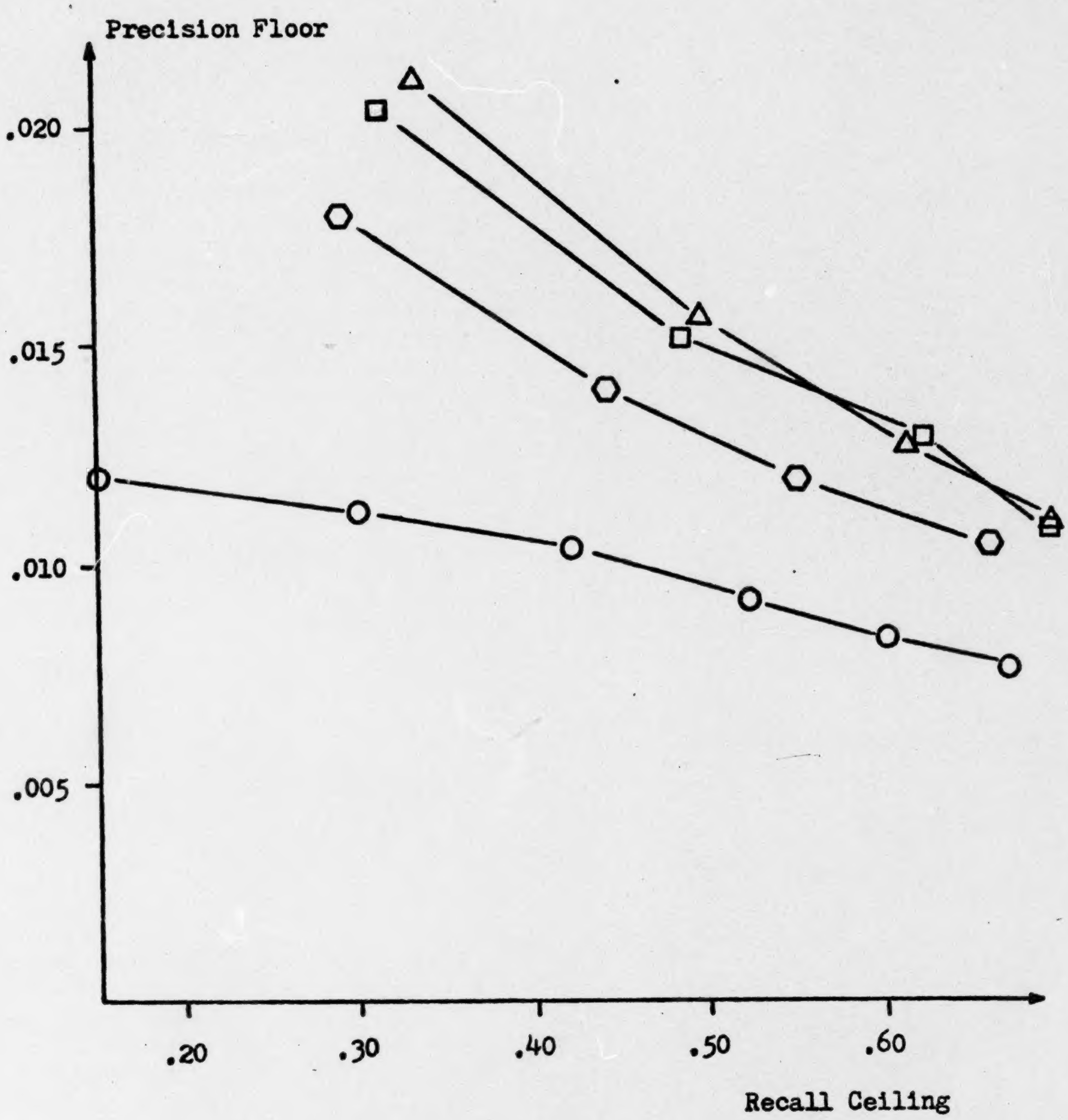


Symbol	Profile Description	MCOS Parameter	E_n
○	P_1 unweighted	τ = 0	.246
⬡	P_1 unweighted	τ = .1	.234
□	P_1 unweighted	τ = .2	.231
△	P_3^* weighted		.214

Behavioral Characteristics of Non-Relevant Profiles
Unweighted Vectors, Size Dependent Cosine Function
Hierarchy 1, Level 1
Figure V-30

contribution of $1/\sqrt{k}$, an amount which is considerably smaller for longer profiles (large clusters) than shorter profiles (small clusters). Thus for the same number of matching profile and query terms, the shorter profile obtains a larger correlation. Furthermore, prior to deletion, all profiles contain about the same set of terms so it is quite probable that several query-profile correlations result in the same number of matching terms. In the case of P_2 or P_3 vectors, these terms are differentiated by their weights; obviously this is not the case for P_1 vectors. Consequently, in usual circumstances, small clusters (hence, short profiles) represented by unweighted profiles usually receive high correlations and are expanded, regardless of their relevancy. If the above conjectures are true, then selective term deletion should improve performance by reducing the occurrence of query-profile matches which involve the same number of terms and by making the values of $1/\sqrt{k}$ more uniform throughout the profile collection.

In order to test this idea, unweighted profiles are made from weighted profiles after deleting unimportant, low frequency terms as described earlier. Specifically, $P_3^*(\delta = -1)$ vectors have the weights of all remaining terms set to a constant value. These unweighted profiles are denoted by $P_1^*(\delta = -1)$. Since each vector contains only those terms which are most characteristic of the corresponding cluster, it is much less likely that correlations involve the same number of matching terms. Figures V-31 and V-32 compare the performance of unweighted profiles with and without such term deletion. Figures V-33 and V-34 contain the data related to their bias behavior. On both levels, a performance improvement and unbiased behavior is noted so the previous conjectures

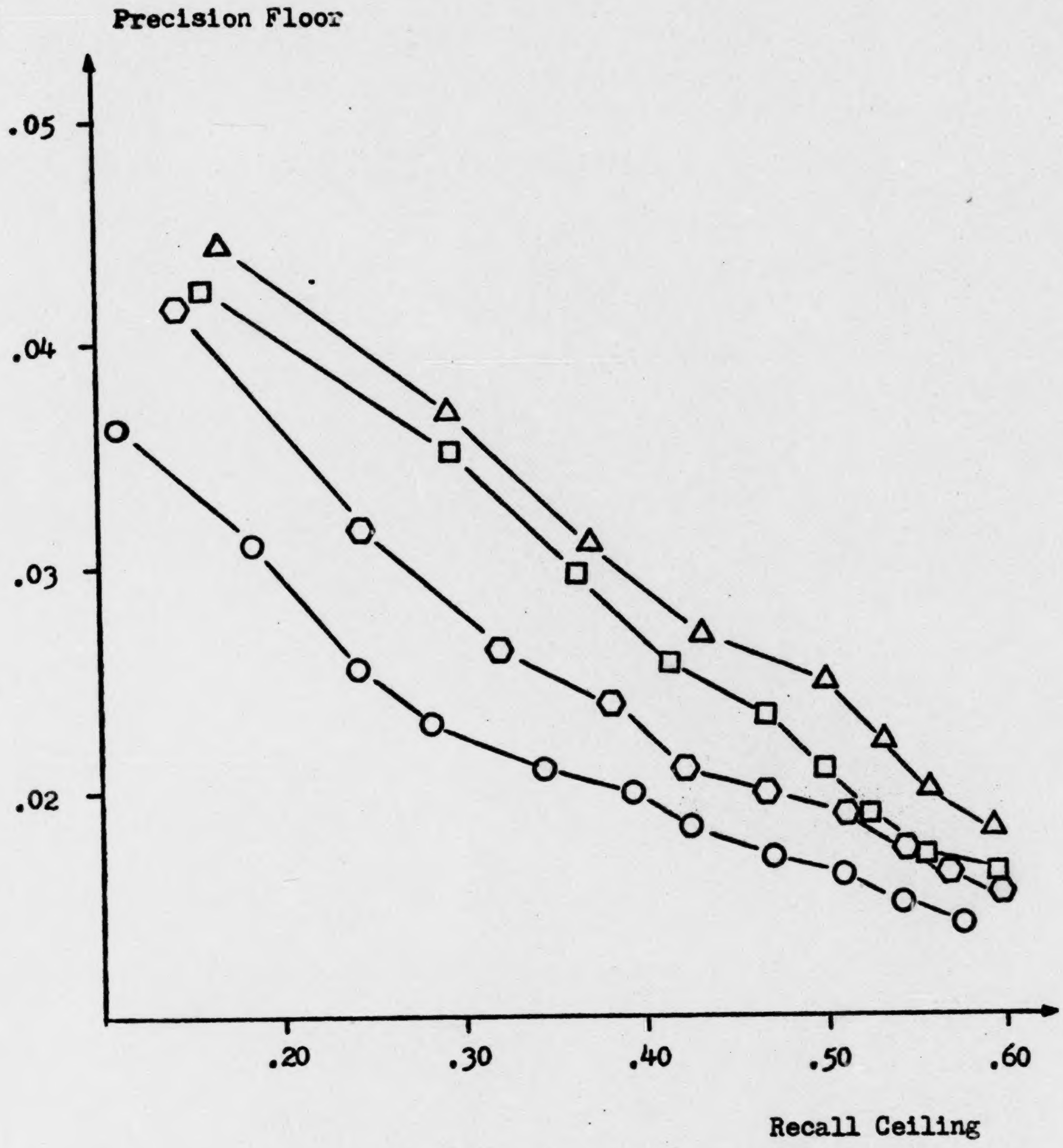


Symbol	Profile Description	Length Ave.	 Std. Dev.
○	P_1 $\delta = -\infty$ (no deletion)	812	200
⬡	P_1^* $\delta = -1$ (from P_3^*)	141	28
□	P_3^* $\delta = -1$	141	28
△	P_3^* $\delta = -\infty$ (no deletion)	812	200

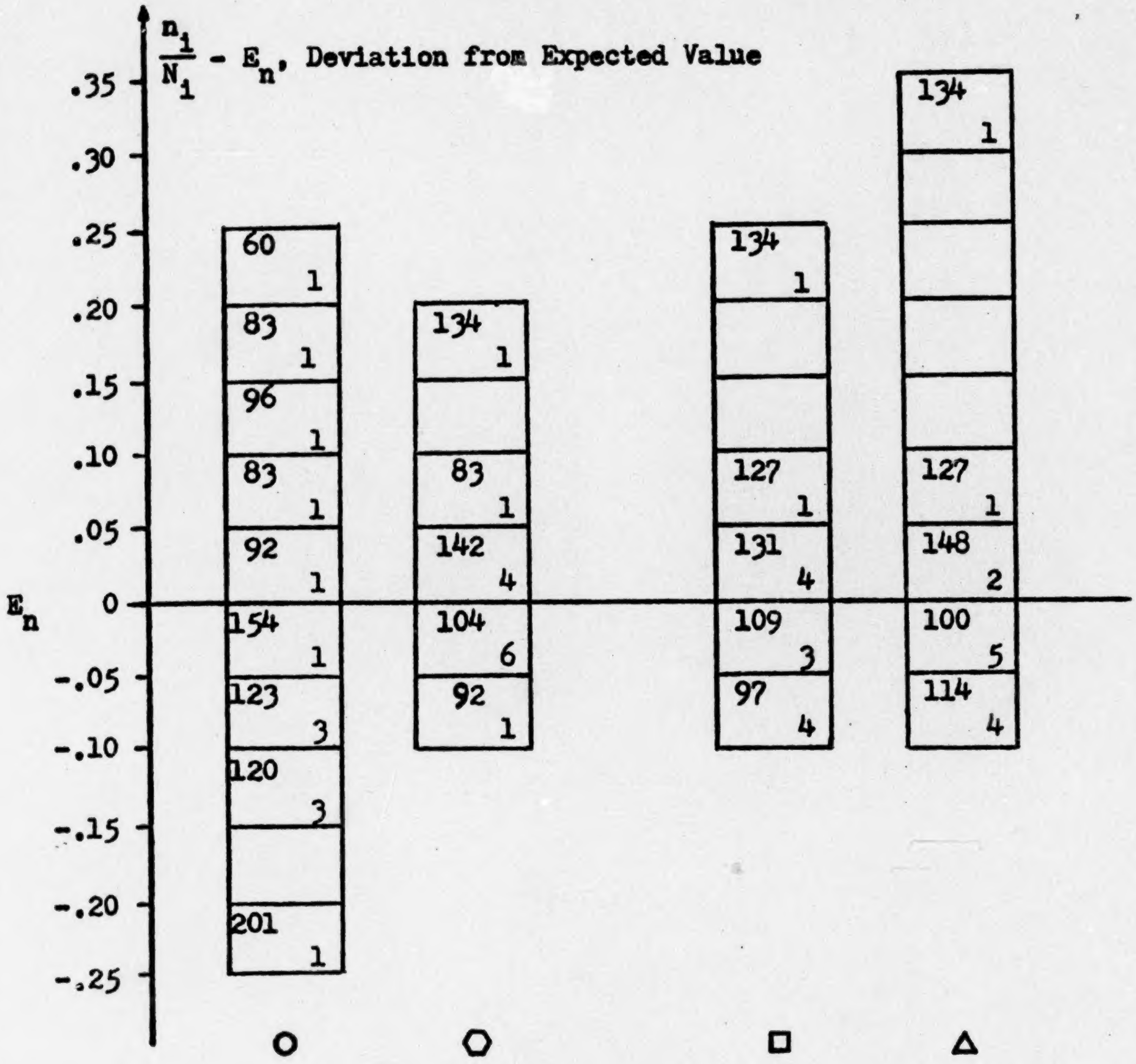
Performance of Unweighted Profiles with Term Deletion

Hierarchy 1, Level 1

Figure V-31



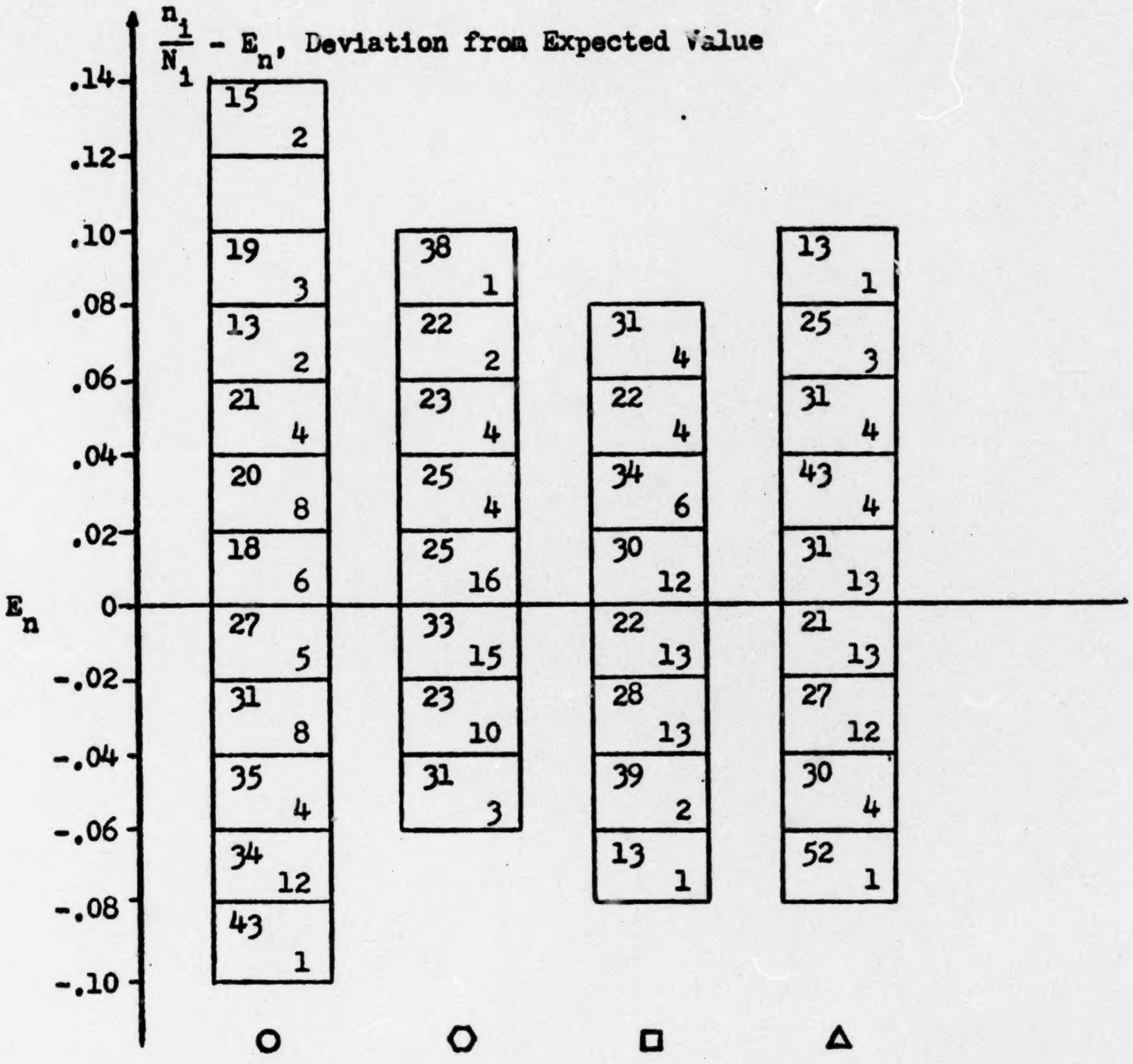
Symbol	Profile Description	Length Ave.	Std. Dev.
○	P_1 $\delta = -\infty$ (no deletion)	323	136
⬡	P_1^* $\delta = -1$ (from P_3^*)	71	23
□	P^* $\delta = -1$	71	23



Symbol	Profile Description	E_n
○	P_1 $\delta = -\infty$	.246
⬡	P^*_1 $\delta = -1$ (from P^*_3)	.205
□	P^*_3 $\delta = -1$	.205
△	P^*_3 $\delta = -\infty$	.214

Behavioral Characteristics of Unweighted Profiles
With Term Deletion
Hierarchy 1, Level 1

Figure V-33



Symbol	Profile Description	E_n
○	P_1 $\delta = -\infty$	.089
⬡	P_1^* $\delta = -1$ (from P_3^*)	.082
□	P_3^* $\delta = -1$	.079
△	P_3^* $\delta = -\infty$	.078

Behavioral Characteristics of Unweighted Profiles
With Term Deletion
Hierarchy 1, Level 2

Figure V-34

appear valid. In fact, the performance difference between the shortened weighted and unweighted vectors (curves for $P_1^*(\delta = -1)$ and $P_3^*(\delta = -1)$) is much smaller than expected. This suggests, perhaps, that fine frequency distinctions among important index terms are much less valuable than selection of good index terms themselves. The selection procedure used here (deletion) is crude and does not attempt to obtain good index terms, but tries to eliminate bad ones. However, if an independent procedure can be found which selects only pertinent terms, then it may be that complex weighting schemes are completely unnecessary.

From the previous experiment one concludes that once noise terms are removed from a document or profile, fine term distinctions based on frequency are not particularly useful. In fact, a few broad weight categories might provide as good a performance as a complete weight range; hence the notion of partial weighting as opposed to the previous full range weighting. In order to test partial weighting, the terms in $P_3^*(\delta = -1)$ vectors are placed in one of four weight classes by the following procedure. Given a profile, new values of μ and σ are computed, and the smallest (MIN) and largest MAX) remaining weights are determined. Bounds on the weight classes are:

Class	Lower Limit	Upper Limit
1	MIN	$(\mu-\sigma)\ \lvert P\rvert$
2	$(\mu-\sigma)\ \lvert P\rvert$	$\lvert P\rvert$
3	$\lvert P\rvert$	$(\mu+\sigma)\ \lvert P\rvert$
4	$(\mu+\sigma)\ \lvert P\rvert$	MAX

Finally each term is assigned a new weight equal to the midpoint of the class indicated by its original weight. Figure V-35 compares the performance of $P_3^*(\delta = -1)$ profiles with full, partial, and no weighting. The results obviously substantiate the usefulness of partial weighting, at least for this hierarchy. This is not surprising considering that

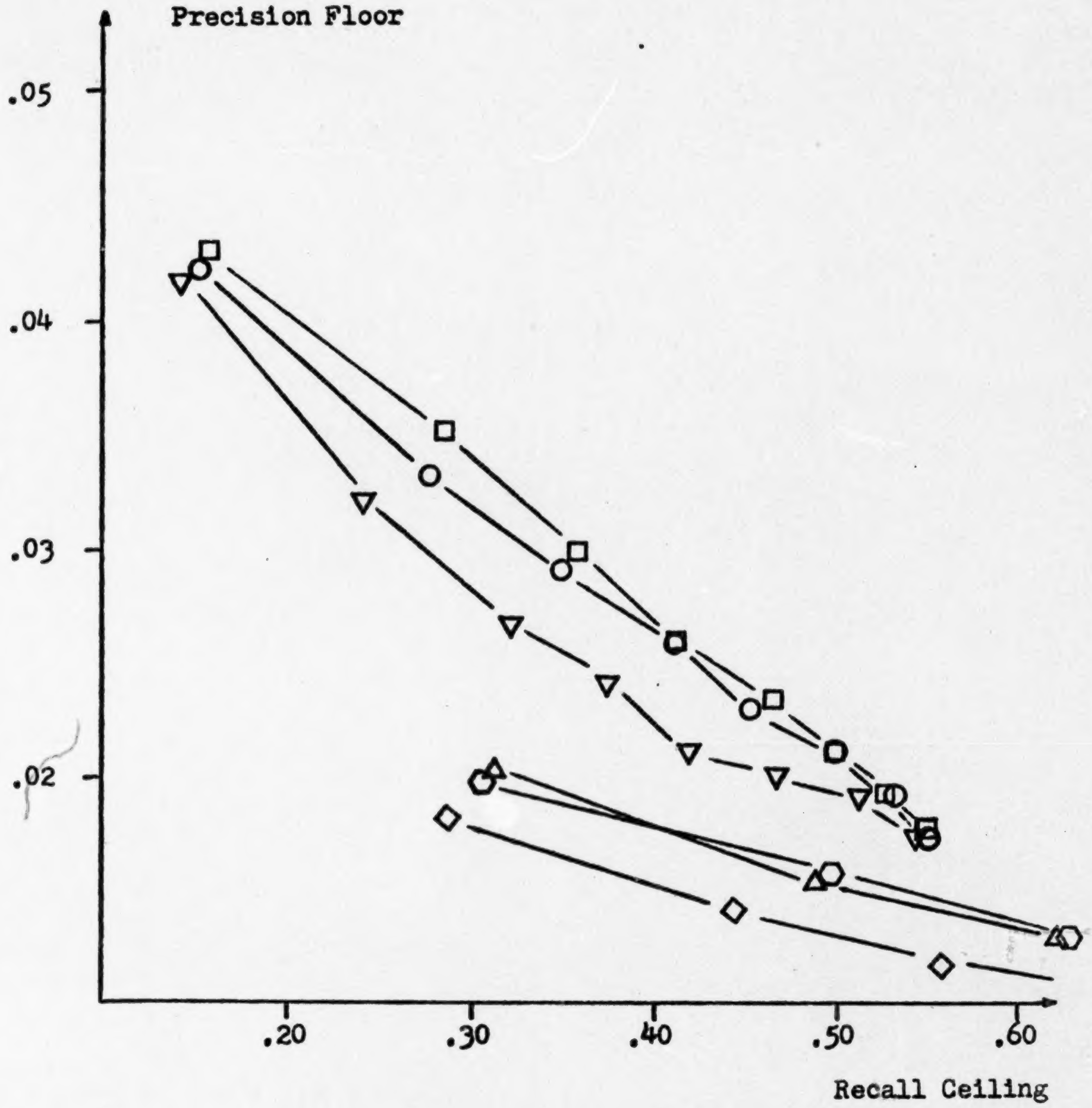
a) the use of weight classes eliminates correlation domination from very high frequency terms and

b) deletion of low weight terms removes many terms that do not play a part in causing cluster formation.

As expected, unweighted vectors provide somewhat poorer performance in spite of the improvements caused by term deletion. More surprising is the fact that only 4 weight classes appear nearly equivalent to a full range of weights. A 2-class scheme (LOW and HIGH) could be expected to provide performance slightly inferior to 4 classes. Two classes would be the easiest to implement in the SMART system since the sign bit of term identifiers (concepts) could denote the weight. An interesting way of producing weight classes in a document or profile might be to assign the weight w to a term of frequency f according to the formula:

$$w = \mathrm{MAX}\left\{0, \lceil \log_x f \rceil - y\right\} \qquad \text{(V-9)}$$

where x and y are constants. The logarithm function smoothes out the weight range (similar to the use of ranks); the ceiling function produces a number of weight classes (1, 2,...); and y acts a deletion cutoff if only terms with positive weights are retained. This scheme appears to produce a vector with all the desirable properties mentioned so far and uses an extremely simple mechanism. Undoubtedly, there are many such schemes.



Symbol	Level	Symbol	Level	Profile Description
△	1	□	2	Full weighting
⬡	1	○	2	Partial weighting (4 classes)
◇	1	▽	2	Unweighted

Performance of Profiles with Full, Partial, and No Weights

Hierarchy 1, $P_3^*(\delta = -1)$ Profiles

Figure V-35

This section describes a set of experiments with unweighted and partially weighted profiles. The results indicate that

a) term deletion in unweighted profiles causes significant performance improvement and reduces their bias with respect to cluster size and

b) profiles with partial weighting (4 classes) can achieve performance equivalent to full range weighting along with some storage economy.

Whether or not a system can take advantage of the efficiency in partial weighting may depend on implementation factors such as machine word size and the number of bits allocated to term identifiers. Otherwise the choice lies at the extremes of no weighting or full range weighting. In either case, the fact that a small number of weight classes (1, 2,...) works so well points out that fine distinctions among term frequencies is not needed for document retrieval.

8. Summary of Results for Hierarchy 1

The descriptions, methods, test procedures, and results presented in the preceding sections all deal with Hierarchy 1 and cluster-oriented evaluation. These are used because the hierarchy contains the number and size of nodes considered typical and because the evaluation is independent of a number of search parameters. The large number of options to be tested make it impossible to examine them using all three hierarchies and both evaluation methods available. At this time, it is appropriate to summarize the findings for Hierarchy 1 and to select several options for more complete testing in the other clustered collections. Below, the findings are listed by section.

A. Standard Profiles

1) Weighted profiles perform significantly better than unweighted profiles, $P_3 > P_1$ and $P_2 > P_1$.

2) Term weights based on document frequency appear equivalent to weights based on total term occurrence, $P_3 \approx P_2$.

3) A slight performance advantage is observed for unweighted profiles made from the shorter P_2 vectors as opposed to those made from the longer P_3 vectors.

B. Rank Value Profiles

1) Base values should be kept small to maintain distinctions among terms; however too small a value biases search results.

2) Use of a minimal weight origin, constant for all profiles, enhances performance, eliminates bias, and avoids the problem of base value selection.

3) Weights based on frequency ranks avoid correlation domination and give better performance than weights based on frequency counts, $P_3^* > P_3$ and $P_2^* > P_2$.

C. Biased Search Results

1) Unweighted profiles and rank value profiles using a large base value show a definite bias in favor of small clusters. Reducing the base value decreases the bias and, to some extent, is accompanied by a performance improvement.

2) P_2, P_3, P^*_2, and P^*_3 vectors show very little bias in their search performance within the same hierarchy level.

3) PF-RC performance improvement has a high correlation with a reduction of bias; however, reducing bias does not necessarily produce an automatic performance improvement.

4) There is no need to include cluster size as a factor in determining query-profile correlations within the same hierarchy level.

D. Profile Length

1) A large number of low frequency terms can be deleted without greatly reducing search performance,

$P^*_3 \approx P^*_3(\delta = -1)$.

2) Major profile terms cannot be selectively deleted nor transfered to a parent profile

$P^*_3(\delta = -1) > P^*_3(\delta = -1, \ = 1)$.

E. Frequency Considerations

Term weights within individual profiles should be non-decreasing with frequency.

F. Unweighted and Partially Weighted Profiles

1) Term deletion improves performance and eliminates bias when using unweighted profiles, $P^*_1(\delta = -1) > P_1$.

2) A limited number of weight classes give performance which is equivalent to using a full weight range.

3) Fine term distinctions based on frequency are of limited appropriateness in gauging retrieval significance.

It is easy to see that a number of findings are related. For example, vector length affects storage considerations and the performance of unweighted and partially weighted profiles. Another example, frequency ranks and weight classes are both techniques for reducing correlation domination. And another, the choice of base value (or weight origin) or the use of unweighted vectors strongly affects the amount of bias in search results.

The profile types selected for use in the confirmation tests on Hierarchies 2 and 3 are P_3, P_3^*, $P_3^*(\delta = -1)$, and $P_1^*(\delta = -1)$. The P_3 vector is a standard profile serving as a basis of comparison. The P_3^* profile showed the best performance of any in Hierarchy 1. The remaining vectors are more economical versions of the P_3^*. All contain those qualities found to be most beneficial thus far: unbiased behavior, lack of correlation domination, and storage economy.

9. Confirmation Tests

To verify the previous results, a subset of the experiments are repeated on Hierarchies 2 and 3. If such confirmation tests yield the same general results, then the conclusions drawn from the earlier experiments are greatly strengthened. As mentioned earlier, four types of profiles are used in the confirmation tests thus making possible to investigate:

a) the superiority of relating profile term weights to frequency ranks rather than frequency counts (P_3^* versus P_3);

b) the ability to delete a large number of low weight profile terms without a large performance loss ($P_3^*(\delta = -1)$ versus P_3^*); and

c) the relative performance of shortened profiles with and without term weights ($P_1^*(\delta = -1)$ versus $P_3^*(\delta = -1)$).

The results of these tests determine, to a large extent, the best profile for searching a clustered hierarchy.

Since Hierarchies 2 and 3 have received little attention, it is appropriate to review their properties as described in Section IV.3 and summarized in Table V-4. Hierarchy 1, used exclusively so far, has low overlap (7%) and document clusters which approximately fill one disk track (28 documents). Hierarchy 2 has high overlap (91%) and document clusters of about the same size. Because of the high overlap and the fact there are only a few nodes on level 1, it is possible to make only very broad distinctions among them. Hierarchy 3 has no overlap and averages 14 documents per cluster. Since there are a moderately large number of nodes per level, the search algorithm should have less difficulty distinguishing relevant profiles than in the other hierarchies. In all cases the shortened profiles ($\delta = -1$) have about 20% of the length of their original vectors.

The confirmation experiments employ both cluster-oriented evaluation (RC-PF data gathered from both levels) and SMART evaluation (P-R data from narrow and broad searches) as described in Section IV.4. Consequently, each of the 4 profiles types is involved in 4 searches in 3 hierarchies, making a total of 48 performance curves. Because of the many variables

Property	Hierarchy		
	1	2	3
Level 1 (Profiles)			
Number of nodes	13	6	28
Average crown	115	446	50
Average sons	4	16	4
Average profile length, P_3 or P^*_3	812	908	526
Average profile length, $P^*_3(\delta = -1)$ or $P^*_1(\delta = -1)$	141 (17%)	207 (23%)	103 (20%)
Level 2 (Profiles)			
Number of nodes	55	94	103
Average crown	27	28	14
Average profile length, P_3 or P^*_3	323	311	197
Average profile length, $P^*_3(\delta = -1)$ or $P^*_1(\delta = -1)$	(22%)	(22%)	(24%)
Level 3 (Documents)			
Number of nodes	1500	2679	1400
Overlap*	7%	91%	0%

*Overlap $\equiv$ ratio of total nodes on level 3 to collection size (1400) less one.

Summary of Hierarchy Properties

Table V-4

involved and the large number of curves, the actual plots are included as a special section (Appendix C). A result summary is contained in Table V-5 which shows the relative merit of each profile in each test case. For example, in Hierarchy 3 and for a broad SMART search, the P^*_3 profiles perform best, and P_3 and $P^*_3(\delta = -1)$ vectors give equivalent performance and share an average rank of $2\frac{1}{2}$, and the $P^*_1(\delta = -1)$ profiles performed poorly. The individual results for each case do not differ greatly from the overall results, thus indicating that the findings are stable and not coincidental. Consequently, it is likely that the effects observed here occur in most other document collections besides the Cranfield. The relative merit of each profile is about the same throughout all hierarchies, so the following summary conclusions can be made.

a) Weights of profile terms should be based on frequency ranks, $(P^*_3 > P_3)$. The use of ranks is an effective way of reducing correlation domination and bias in search results.

b) A large number of low weight profile terms can be deleted without a large performance loss, $(P^*_3 > P_3 \approx P^*_3(\delta = -1)$. For the chosen deletion parameter, the performance loss from using about 20% of the original terms is about the same as the performance gain made by switching term weights to ranks. Whereas this amount of deletion is probably not optimal, it does indicate that a large length reduction does not lead to disastrous search results.

Hierarchy and Profile Type	Cluster Evaluation*		Smart Evaluation*	
	Level 1	Level 1	Narrow Search	Broad Search
Hierarchy 1				
P_3^*	1	1	1	1
P_3	3	3	3	2
$P_3^*(\delta = -1)$	2	2	2	3
$P_1^*(\delta = -1)$	4	4	4	4
Hierarchy 2				
P_3^*	2	1	1	1
P_3	2	2½	2	2
$P_3^*(\delta = -1)$	2	2½	3	3
$P_1^*(\delta = -1)$	4	4	4	4
Hierarchy 3				
P_3^*	1	1	1½	1
P_3	2½	2	1½	2½
$P_3^*(\delta = -1)$	2½	3	3	2½
$P_1^*(\delta = -1)$	4	4	4	4
Average Rank				
P_3^*	1.1			
P_3	2.3			
$P_3^*(\delta = -1)$	2.5			
$P_1^*(\delta = -1)$	4.0			

*Entries denote merit in terms of rank: first, second, etc.
Ties are given the average rank.

Relative Merit of Selected Profiles in Confirmation Tests

Table V-5

c) Shortened unweighted profiles performed poorly in every case and should not be used, $P^*_1(\delta = -1) > P^*_3(\delta = -1)$. However, the previous sections suggest that complex weighting schemes may not be necessary.

In general, the findings for Hierarchy 1 are substantially confirmed by these tests, a possible exception being the case of unweighted vectors. In the initial collection the $P^*_1(\delta = -1)$ profiles gave promising performance (still low), which did not show itself in the other hierarchies.

What is missing from this discussion naturally, is the description of just how profiles are ranked on the basis of relative merit and what constitutes a significant difference in performance. A detailed discussion of these problems is reserved for Appendix C, but a summary is as follows. Generally a 2%-4% difference in measures (PF-RC, P-R, NR, NP, etc.) is considered significant. This is about half the amount used in earlier SMART experiments. However since 4 times as many requests are involved here, conclusions have about the same level of confidence. In cluster-oriented evaluation, the PF-RC curves are compared on a point-to-point basis and judgment rendered on the merit of each profile type. Since the curves are generally parallel, this technique poses no problems. SMART's precision-recall plots and accompanying normalized measures present some difficulty because the number of document and profile correlations differ among searches. At times it is necessary to determine whether performance should be traded for cost (fewer comparisons). The following criteria are used in these circumstances:

a) the most desirable profile is that giving superior search performance for the least effort;

b) for the same number of correlations on each level, merit is determined directly from P-R values and the normalized measures;

c) profile correlations weigh much more heavily than document correlations in determining "equal effort"; and

d) a 2% difference in normalized measures or a 4% difference in P-R curves is considered significant and is offset only by substantially less search effort (one or fewer profile correlations).

Using this procedure, the relative merit of each profile type is obtained for each hierarchy and the entries in Table V-5. This data, then, leads to the summary conclusions stated earlier in this section.

10. Discussion

This chapter presents a long series of experiments related to profile construction. In the process, new analysis and evaluation methods are developed which have applications to other studies as well. The results of the initial experiments are adequately summarized in Section 8; Section 9 summarizes the confirmation tests. Here, it is sufficient to say that techniques have been developed for constructing an adequate and economical profile for a clustered document collection.

This does not imply that there is no room for improvement. If the best precision-recall curves are compared with a similar curve for a full search (Figure IV-5), cluster searching appears to give very poor performance. However, it must be said that cluster searching is not designed for high recall work, but for flexibility and cost-performance tradeoff. Still, a comparison of the best PF-RC curves with the best achievable results for the same collections (Figures IV-7 and IV-8) show that profiles, search strategies, and correlation functions could be greatly improved. Consequently, under the current situation, search strategies should be designed on a minimum exclusion principle. That is, only nodes with very low correlations should be excluded from consideration while the rest are expanded. This contrasts to the current minimum inclusion philosophy which expands as few nodes as possible to provide a user with his requested number of documents. The former procedure results in greater cost, but more effective retrieval. Hopefully on larger collections (100,000 items), 90% of the documents could be easily excluded while the rest require detailed examination.

References

1. L. B. Doyle, Breaking the Cost Barrier in Automatic Classification, SDC Paper SP-2516, July 1966.

2. B. Litofsky, The Utility of Automatic Classification Systems in IS&R, Doctoral Thesis, University of Pennsylvania, 1968.

3. H. P. Luhn, The Automatic Creation of Literature Abstracts, IBM Journal of Research and Development, April 1958.

   Paper also reported in: C. T. Meadow, The Analysis of Information Systems, John Wiley & Sons, New York, 1967.
   and in T. C. Lowe, D. C. Roberts, On-Line Retrieval, RADC Technical Report TR-69-304 from Informatics, Inc., November 1969.

4. K. Bonwit, J. Aste-Tonsman, "Negative Dictionaries," Report ISR-18 to the National Science Foundation, Department of Computer Science, Cornell University, 1970.

Chapter VI

File Maintenance Experiments

1. Introduction

File maintenance (or updating) is the process by which new documents are added to the data base, including whatever re-organization is required to maintain standards for search time, storage economy, and quality of retrieved output. A good updating procedure is especially important in a clustered collection in order to prolong the useful life of the document classification. However, Chapter III points out that no hierarchy can stand unlimited growth without changes to its profiles and structure. One part of file updating, then, is the alterations to be made to individual profiles. Presumably, altering a profile shifts it to a position within the cluster which more adequately represents the combination of new and old documents. These experiments examine five alteration procedures for each profile on a document update path:

Original Profile Type

Maintenance Procedure	Weighted, $P_3^*(\delta = -1)$	Unweighted, $P_1^*(\delta = -1)$
Construct completely new profiles	X	X
Alter weights of only existing profile terms	X	
Use existing profile (add document to crown only)	X	X

The purpose of the first study in this chapter is to determine which profile maintenance schemes are most effective.

At some point, file maintenance requires changes to be made in the hierarchy structure because new documents alter the character of the collection and ultimately cause the original classification to lose its value. For example, clusters may become too large or polarized and cannot be represented adequately by a single profile. Or many additions in one part of the collection might indicate a more logical classification would split various clusters and combine their parts in a different way. Without re-structuring (re-clustering), the hierarchy degenerates quite apart from any changes made to profiles. The purpose of the second study is to determine how quickly the retrieval quality (precision-recall) diminishes as the file grows. The rate of hierarchy degeneration is important because it determines the interval between full or partial re-clustering of the document collection.

The experimental approach is to divide the document collection into two groups. One part is clustered and the other is used to update this "original" hierarchy. After updating, the query collection is processed through the combined collection while recording performance statistics (PF-RC, P-R, etc.). Since each request has the potential of recovering all relevant items, performance differences are directly due to the updating scheme and to the proportion of the collection in the updating group. If the size of the updating group is held constant, then the relative value of the profile maintenance procedures can be studied. Choosing a single maintenance procedure and varying the size of the updating group shows how the quality of the hierarchy changes with the addition of new items. The test results show the superiority of using weighted profiles, but indicate very little difference among the various profile maintenance procedures. The policy of modifying the weights of only existing profile terms appears to be a good compromise in this regard. Finally, a clustered collection may increase 25%-50% before sufficient hierarchy degeneration occurrs to require re-clustering.

2. Method

The following experiments use two partitions of the Cranfield 1400 documents: one separating documents into two equally sized sets referred to as A and B, and another which divides B into the halves C and D. The result is four collections for testing file maintenance procedures:

Clustered Collection		Update Collection	
Set	% of Total	Set	% of Total
A∪B	100%	∅	0%
A∪C	75%	D	25%
A	50%	B	50%
D	25%	A∪C	75%

The partitions deliberately maximize the number of queries affected by updating. For example, if 75% of the total Cranfield collection is to be clustered (set A∪C), then to the extent possible, 75% of the relevant for each request are placed in that set, leaving 25% of the relevant for each request in the updating set. Consequently, searches take full advantage of the large number of queries and reliably show the consequences of file updating. In particular, the variance of behavior among the queries is greatly reduced.

The collection partitioning algorithm is a manual procedure for dividing a set of documents into two non-overlapping sub-sets, each containing half the relevant for each query. Initially, the scheme is applied to all 1400 documents to generate sets A and B and then to set B to obtain subsets C and D. Actually, the algorithm considers only relevant documents and splits the non-relevant items afterwards. To control overall characteristics, queries are processed in order of decreasing number of relevant documents. Given a specific query, counts are made to determine how many of its relevant are already assigned to each partition set. The remaining relevant are assigned so that

a) for the entire query, all relevant are split evenly into each partition set;

b) relevant documents with consecutive numbers are not assigned to the same set; and

c) the cumulative number of items in each partition set is approximately the same.

Condition b is required since the Cranfield collection is arranged in subject order (somewhat). Condition c takes care of requests with an odd number of relevant. At times, previous assignments must be reworked to accommodate new queries; the frequency of these "backtracking" instances is reduced because of the order of query processing. In the partitions made for the test experiments, about 2% of the total assignments could not be made by the above criteria without extensive, prohibitive backtracking. Even though "erroneous" assignments are made in these cases, the collections are sufficiently accurate for their intended purpose. Appendix B lists the members of sets A, C, and D.

The classification parameters (Dattola's clustering algorithm) for these experiments are designed to produce clusters similar to those of Hierarchy 1 in the previous chapter. A reasonably constant cluster size is maintained to provide comparable data collections and to simulate adherence to an operational guideline of holding to an optimal cluster size (if an optimum were actually known). These particular parameters yield clusters of moderate size and overlap and are considered suitable for larger files. Hierarchy 1 represents a collection without updating; its performance statistics are those which would be obtained if an updated collection were freshly clustered using the chosen parameters. The hierarchies generated from the other sets are subjected to various amounts of updating. In both cases nodes are characterized by two types of profiles: $P_3^*(\delta = -1)$ vectors for weighted updating and $P_1^*(\delta = -1)$ vectors for unweighted updating. Shortened profiles are selected because of their smaller storage requirements. Otherwise the profiles are those giving the best performance for weighted and unweighted vectors. Table VI-1 compares the properties of the four clustered collections before updating occurs. In general, the goal of having hierarchies with similar characteristics is achieved, allowing for the sizes of the collections involved. The most unfortunate difference is the amount of overlap, being considerably higher for Hierarchy 4.

The update procedure for each document begins by determining its update path, that is, the best matching node on the first hierarchy level and the best matching node among its sons. As before, the cosine function is used for matching. If no profile alterations are involved, the new

Property	Hierarchy			
	1	4	5	6
Cluster set	AUB	C	A	D
Percent of original Cranfield	100%	75%	50%	25%
Collection size	1400	1050	700	350
Level 1 (Profiles)				
Number of nodes	13	9	6	3
Average crown	115	133	128	118
Average profile length	141	149	148	150
Average number of sons	4	4	4	4
Level 2 (Profiles)				
Number of nodes	55	36	24	12
Average crown	27	38	34	30
Average profile length	70	86	82	83
Level 3 (Documents)				
Number of nodes	1500	1367	81	360
Overlap	7%	30%	17%	3%

Properties of the Original Clustered Collections Before Updating

Table VI-1

document is simply associated with the selected nodes as described in Chapter III. If profiles are modified or re-constructed, there are several options to exercise. In occasional updating or in real-time operation, profiles are modified each time a document is processed. If used in these experiments, profile changes and search results would depend on the order in which documents are added. Instead, a batch update mode is considered in which the update paths are determined for all new items and all profiles are altered afterward. Because each profile is changed only once, the final hierarchy configuration is independent of the order in which documents are added. The extent to which these experiments predict behavior over many smaller batch updates is unknown. A few documents would probably join different clusters, however the number of such instances is expected to be small. If profiles are completely re-made after updating, processing simply follows the rules laid down in Chapter V. That is, the document vectors in a node's crown are appropriately combined and perhaps re-weighted; and a term deletion cutoff is applied. Such vectors represent the best profiles that can be constructed for the updated hierarchy. In general, new profiles are longer than their previous versions and cause fragmentation of disk storage since the new vectors cannot exactly overwrite their predecessors. In the case of weighted profiles, there is another file maintenance option which alters the weights of only the existing profile terms. Since no new terms are added, the vectors maintain their original length and can overwrite their predecessors. Specifically, consider a set of update documents $U = \left\{D_1, D_2, \ldots, D_x\right\}$ for a node whose profile is $P_3^*(\delta = -1)$. The updated profile is

$$H_3^*(\delta = -1) = P_3^*(\delta = -1) \oplus \sum D_1 \qquad \text{(VI-1)}$$

The operator $\oplus$ denotes normal component-wise vector addition, but limited to only non-zero elements of the left-hand operand. The resulting profile is a hybrid in that it combines the weights of a P_3^* vector (based on frequency ranks) with the term weights of $\sum D_1$ (summed frequency counts). There is no easy way to remove the hybrid weighting property without producing a completely new vector. On one hand, $\sum D_1$ could be converted to use rank weighting and then added to the original profile using the $\oplus$ operation. The result is still not a true P_3^* vector. Instead of a partial solution such as this, the complete hybrid is used in this study. Some consequences of this choice are discussed in Section VI.3.

Because the experiments are conducted with weighted profiles first and then with unweighted profiles, there are two different versions of the updated collections. Table VI-2 compares the properties of the two final, updated collections and shows that

a) the average new profile length differs little in the two collections;

b) there are slightly fewer relevant nodes when weighted profiles are involved; and

c) there is 62%-76% agreement on the first node of the update path and 47%-60% agreement on the complete path.

Comparing collections before and after updating reveals significantly longer profiles (new) and a lower percentage of overlap. The reduction in overlap is due to the fact that new documents are associated with only one node on each level. A fact not shown in the table is that the amount

Property	Hierarchy			
	1	4	5	6
Update set	0	D	B	C
% of original Cranfield	0%	25%	50%	75%
Level 1 (Profiles)				
Number of nodes	13	9	6	3
Average crown	115	172	245	468
Weighted updating				
Average new profile length	141	158	185	245
Average relevant nodes	3.9	3.5	2.8	2.0
Unweighted updating				
Average new profile length	141	156	185	
Average relevant nodes	3.9	3.5	2.9	2.1
Agreement in update path (first node only)		62%	71%	76%
Level 2 (Profiles)				
Number of nodes	55	36	24	12
Average crown	27	48	63	117
Weighted Updating				
Average crew profile length	70	86	105	118
Average relevant nodes	5.3	5.3	4.4	3.4
Unweighted Updating				
Average new profile length	70	89	99	127
Average relevant nodes	5.3	5.4	4.6	3.5
Agreement in update path (both nodes)		47%	52%	60%
Level 3 (Documents)				
Number of nodes	1500	1727	1519	1410
Overlap	7%	23%	8%	1%

Properties of the Updated Collections

Table VI-2

of increase in cluster size varies considerably about its expected value for each hierarchy. In other words, the update scheme causes some clusters to receive many additions, some to experience moderate growth, and others to have only a few changes. However, the amount of growth varies more widely if unweighted profiles are involved. In all cases, increases in cluster size are not skewed toward many large increases and a few small increases or vice versa. Instead the distribution appears uniform, but with a large standard deviation. Uniform growth is not completely unexpected because of the way the collection is partitioned, but it is not a direct consequence of the partition either.

3. Profile Maintenance Procedures

As new documents are added to an existing hierarchy, the nature of clusters changes also. That is, they contain different information or heavier concentrations of older information, etc. A logical move, then, is to alter cluster profiles to reflect this change in character. The purpose of the experiments in this section is to determine which profile maintenance procedure is most beneficial. Hence, the constant quantities are the amount of file updating and the assignments of particular new documents to clusters within either the weighted (WTD) or unweighted (UNWTD) profile hierarchies. The variable quantity is the method of profile alteration itself; several schemes are discussed in Sections III.5 and VI.1 and are denoted as follows:

NEW - make completely new profile vectors, adding new terms, re-weighting, etc.;

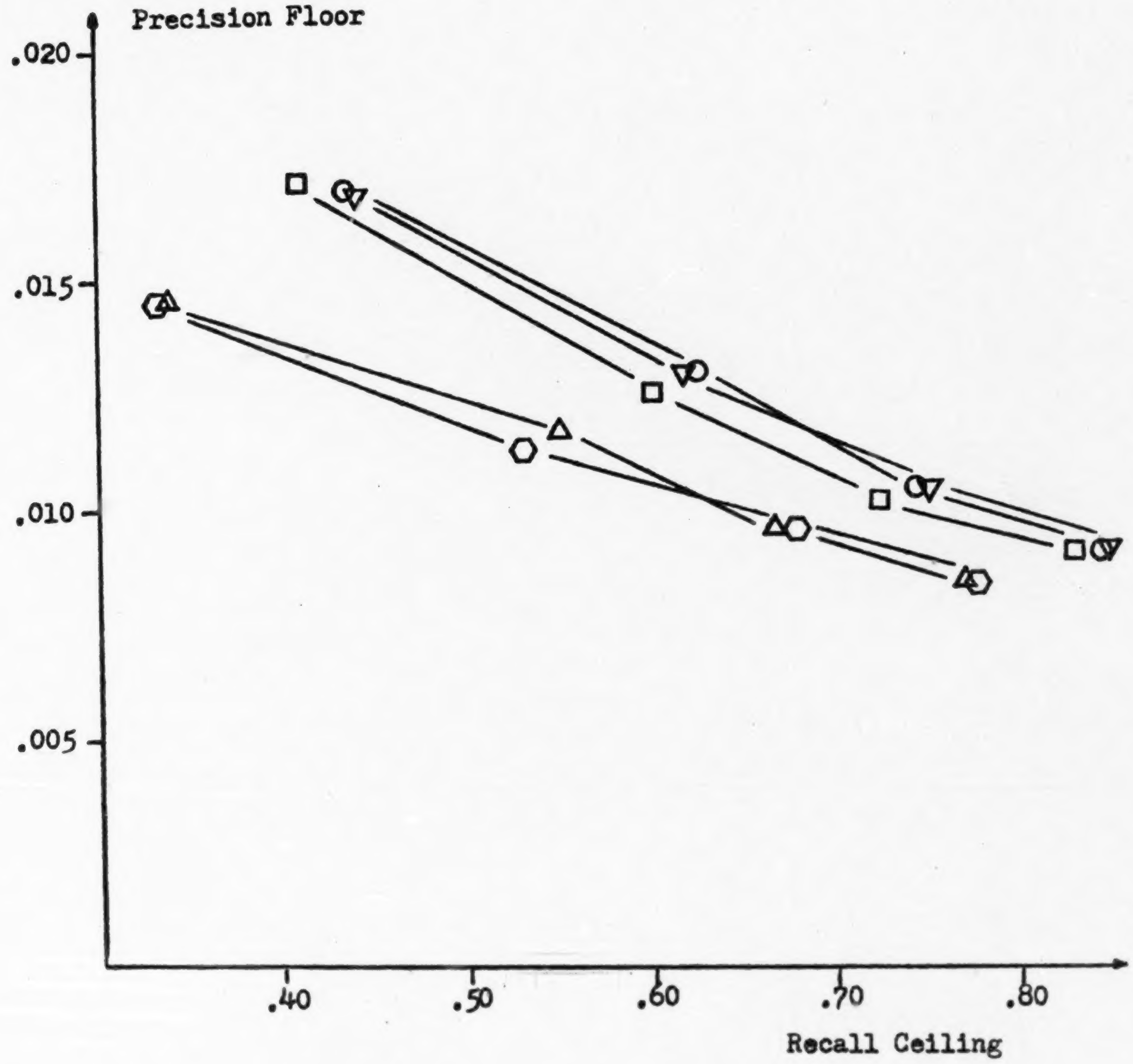
ALTER - alter weights of existing profile terms only (i.e., produce hybrid vectors); and

NONE - no change to the existing profile terms.

In all cases, new documents are properly linked to the hierarchy so that all items are retrievable during searches. Figures VI-1 to VI-5, report the precision floor and recall ceiling statistics for searches using these profile maintenance procedures in the various hierarchies. Only level 1 of Hierarchy 6 is omitted because it contains only two independent data points. On each level, the results are amazingly consistent. Obviously updating a collection with weighted profiles is superior to updating a collection with unweighted profiles, just as predicted by the distribution of relevant clusters (Table VI-2). This finding closes the case against the use of unweighted profiles in almost any circumstances. Here, the unweighted updated collections perform poorly regardless of whether the old profiles or completely new unweighted profiles are used.

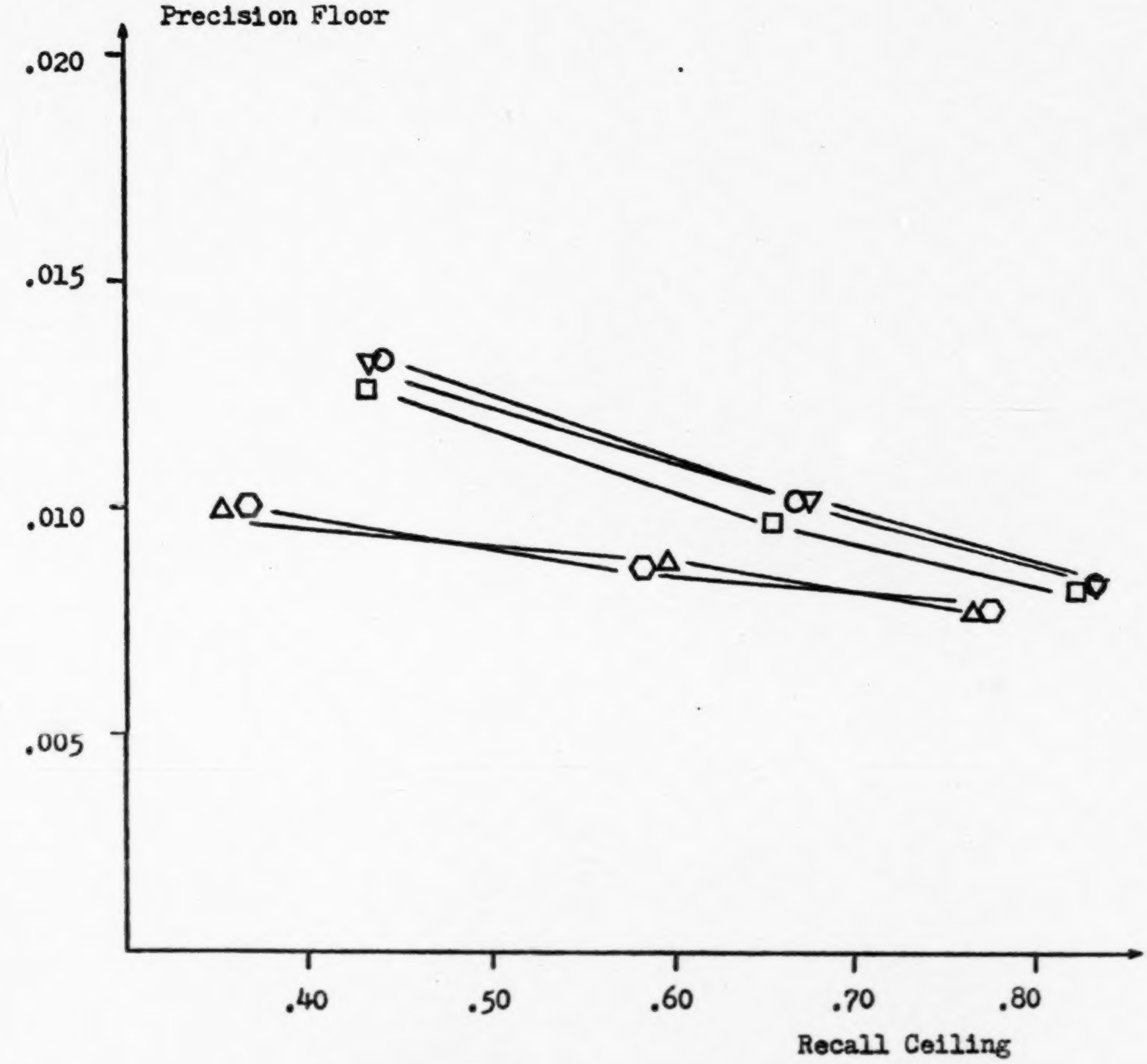
Considering just the weighted profiles, the new (NEW) and hybrid (ALTER) vectors provide nearly equivalent performance even though the new vectors are often substantially longer. In most cases the additional terms are those just below the $\delta = -1$ cutoff in the original profiles before any updating occurs. Hence, the terms have only a marginal effect on performance. It is quite advantageous that the hybrid and new vectors are nearly equivalent since

a) NEW vectors represent the best reasonable profiles that can be made for an updated collection and



Symbol	Maintenance Procedure	Profile Type	Average Length
△	NEW	WTD $P^*_3(\delta = -1)$	158
○	ALTER	WTD $P^*_3(\delta = -1)$	149
□	NONE	WTD $P^*_3(\delta = -1)$	149
△	NEW	UNWTD $P^*_1(\delta = -1)$	156
⬡	NONE	UNWTD $P^*_1(\delta = -1)$	149

Comparison of Profile Maintenance Procedures
Hierarchy 4, Level 1, 25% Updating
Figure VI-1

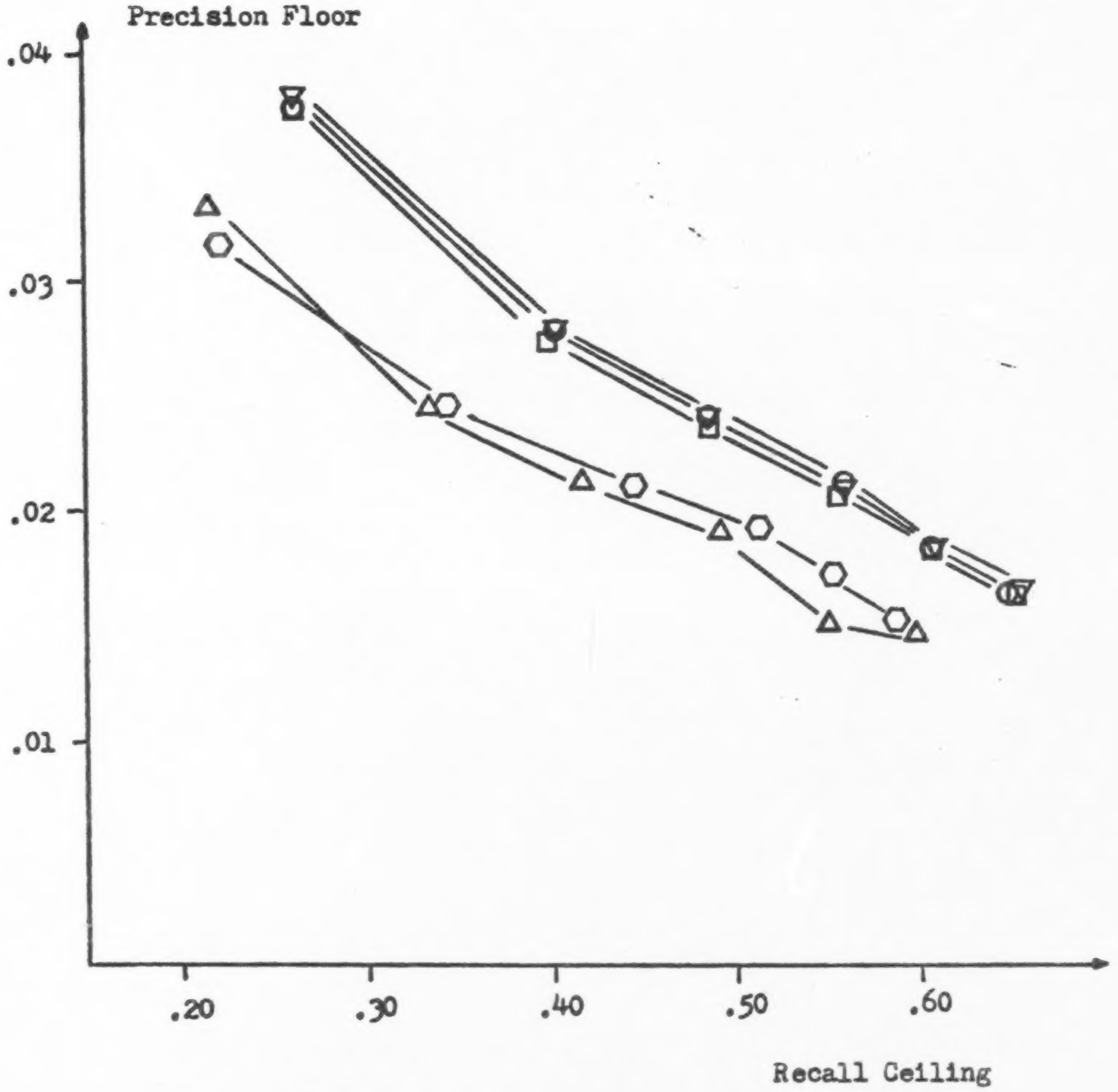


Symbol	Maintenance Procedure	Profile Type	Average Length
▽	NEW	WTD $P^*_3(\delta = -1)$	185
○	ALTER	WTD $P^*_3(\delta = -1)$	148
□	NONE	WTD $P^*_3(\delta = -1)$	148
△	NEW	UNWTD $P^*_1(\delta = -1)$	185
⬡	NONE	UNWTD $P^*_1(\delta = -1)$	148

Comparison of Profile Maintenance Procedures

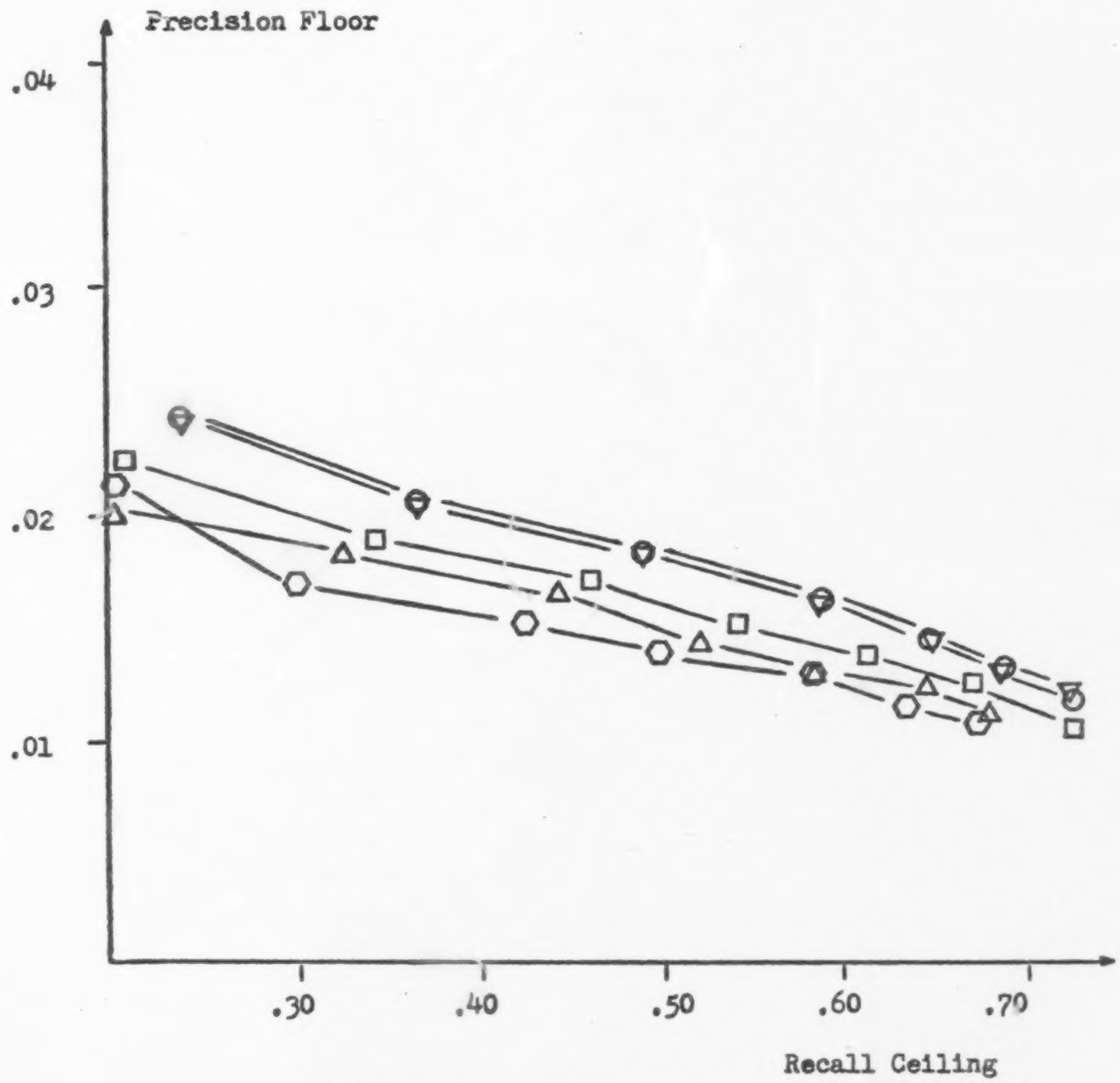
Hierarchy 5, Level 1, 50% Updating

Figure VI-2



Symbol	Maintenance Procedure	Profile Type	Average Length
▽	NEW	WTD $P^*_3(\delta = -1)$	86
○	ALTER	WTD $P^*_3(\delta = -1)$	86
□	NONE	WTD $P^*_3(\delta = -1)$	86
△	NEW	UNWTD $P^*_1(\delta = -1)$	89
⬡	NONE	UNWTD $P^*_1(\delta = -1)$	86

Comparison of Profile Maintenance Procedures
Hierarchy 4, Level 2, 25% Updating
Figure VI-3

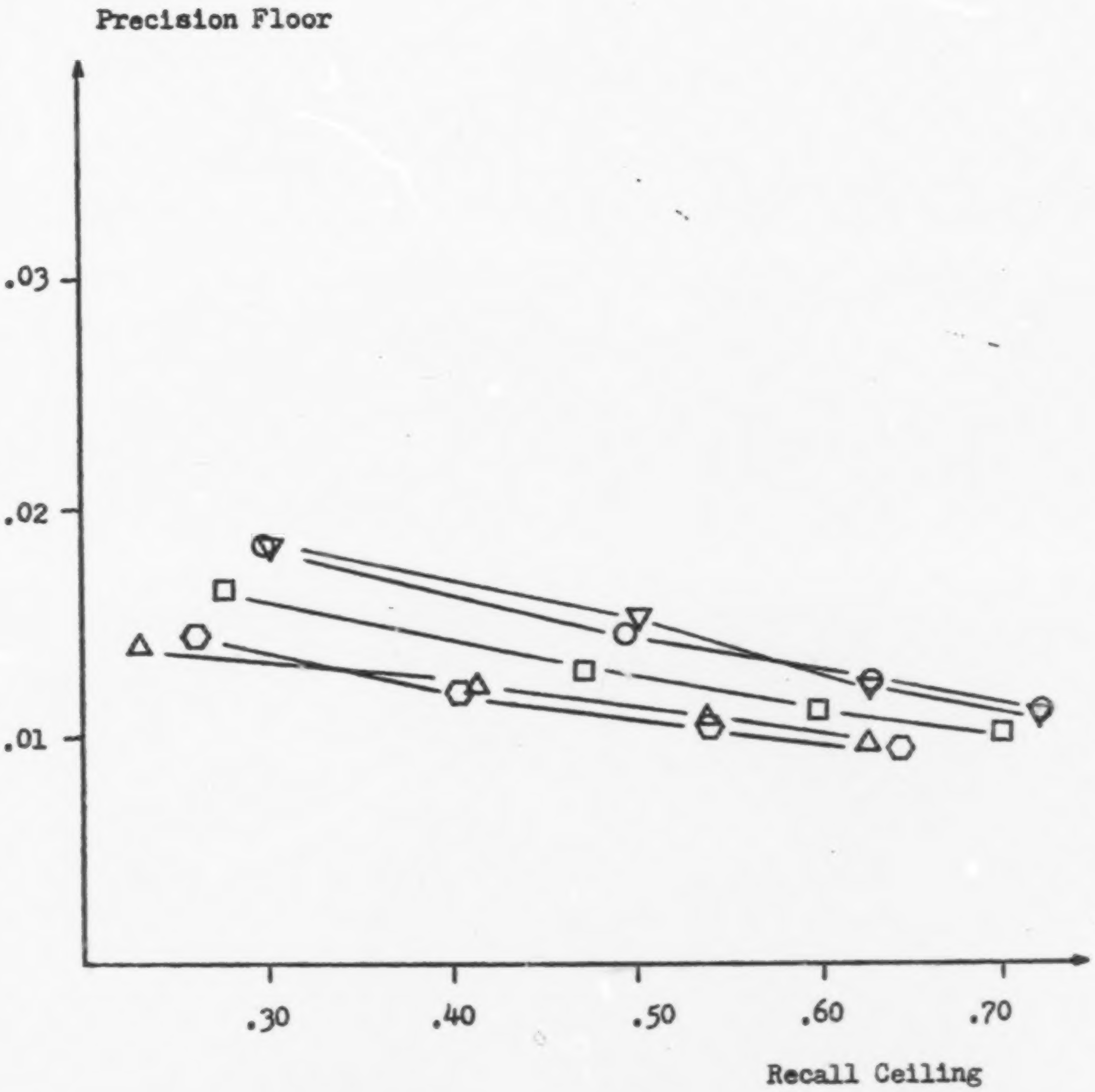


Symbol	Maintenance Procedure	Profile Type	Average Length
▽	NEW	WTD $P_3^*(\delta = -1)$	105
○	ALTER	WTD $P_3^*(\delta = -1)$	82
□	NONE	WTD $P_3^*(\delta = -1)$	82
△	NEW	UNWTD $P_1^*(\delta = -1)$	99
⬡	NONE	UNWTD $P_1^*(\delta = -1)$	82

Comparison of Profile Maintenance Procedures

Hierarchy 5, Level 2, 50% Updating

Figure VI-4



Symbol	Maintenance Procedure	Profile Type	Average Length
▽	NEW	WTD $P_3^*(\delta = -1)$	118
○	ALTER	WTD $P_3^*(\delta = -1)$	83
□	NONE	WTD $P_3^*(\delta = -1)$	83
△	NEW	UNWTD $P_1^*(\delta = -1)$	127
⬡	NONE	UNWTD $P_1^*(\delta = -1)$	83

Comparison of Profile Maintenance Procedures

Hierarchy 6, Level 2, 75% Updating

Figure VI-5

b) ALTER profiles maintain their original lengths and reduce fragmentation of disk space since they can overwrite their predecessors.

Consequently, use of the hybrid (ALTER) profiles is preferable in actual retrieval systems.

Surprisingly, hybrid vectors do not seem to experience correlation domination. Since their term weights are based partially on frequency ranks (from the original profile before updating) and partially on frequency counts (summed over the new documents), some domination could occur when a node experiences a large number of additions. Under these conditions, some terms of its profile have their weights considerably increased. However as suggested in Section V.4, domination involves only a few terms with very high weights and here the use of ranks in the original profile may reduce weights enough so that domination is not observed. A less likely explanation holds that all terms have their weights increased in proportion to their original values so that contribution ratios remain roughly constant throughout the collection.

Weighted profiles which remain unaltered after updating (NONE option) perform slightly less well than NEW profiles for modest amounts of updating (25%) and less well otherwise. This demonstrates the suitability of an update procedure which makes no changes to the profile terms at all. The effect of no alteration is an important consideration in the use of partially weighted profiles, for example, (see Section V.7) in which term weights cannot be altered without destroying the entire vector. In all cases, updating requires some changes to one or more profiles in order to link new documents to upper hierarchy levels. Since these vectors must be re-written anyway, there is little extra savings from not reweighting terms on at least the lowest level where at all possible. As mentioned above, an exception to this is partially weighted profiles.

Finally because weighted profiles maintained under the NEW and NONE options do not differ widely, it is apparent that profiles do not need a great ability to move about their clustered document subspace in order to characterize new items. Some movement seems advisable, but not a great deal is required. This may be related to the fact that the partitioning and update schemes result in more or less uniformly distributed increases in cluster size. Even so, the assumption of random subject acquisition over the entire collection still renders the present results applicable to practical situations since bulk additions in a single subject area can be viewed as random acquisition in a subtree of the original hierarchy.

A summary of the findings in these experiments includes the following.

a) Weighted profiles are superior to unweighted profiles with respect to updating, in that they result in fewer relevant clusters and earlier searching of these clusters.

b) Considering just the use of weighted profiles, there is little difference among the maintenance options NEW, ALTER, and NONE. The last option--no changes to profiles--is slightly inferior for a large number of additions to the file.

c) The simplest and most reasonable update procedure is that of making hybrid profiles (ALTER option). That is, the weights of existing profile terms are adjusted by equation

VI-1, but no terms are added. In addition to providing good performance, these vectors retain their original size. The experimental results presented here are given in terms of RC-PF performance curves. SMART precision-recall plots for 18 narrow and broad searches using these profiles may be found in Figures VI-6 to V-12 in the next section. These curves are not repeated here because they provide no information that changes the above conclusions.

Recent work by Kerchner (3) in this area substantiates these findings. This work uses a different hierarchy, employs slightly different update procedures, considers a 50% update fraction, and uses a single search strategy with one iteration of relevance feedback.

4. Degeneration of the Hierarchy

A document classification bases its groupings on the data available at a single moment of time. Afterward, new items are blended into the existing structure. In general, updating causes a reduction of search speed since new documents may be stored in overflow areas away from the rest of their cluster. Indexed sequential access schemes use a number of techniques to handle overflow records such as storing them in the same cylinder or pack; writing blocked or unblocked records; and using dynamic cylinder reorganization (1, 2). Since SMART simulates cluster searches, it is impossible to measure the exact increase in search time due to updating. A rough approximation suggests that a full disk rotation might occur between inputs of document records (unblocked) in the overflow area. In any case, search speed can be increased simply by re-writing the file in correct physical sequence. Although there is expense involved in this solution, it involves only data movement and not a structural re-organization of the file.

In addition, new documents subject a hierarchy to a subtle form of degeneration that ultimately necessitates partial or full re-clustering. The problem stems from the fact that new documents alter the character of the collection and individual clusters so that the original classification loses its value. Regardless of whether updating includes alterations to profiles, clusters may become polarized or very similar in content (see Figure III-5). As a result, users receive poorer output from searches because profiles no longer accurately represent all the documents in their clusters and because the classification is no longer a "logical" partition of the collection. The solution to this problem is some form of re-clustering. The experiments in this section attempt to determine how frequently re-clustering must occur as a function of file growth. Consequently, a consistent profile maintenance scheme is used and the amount of updating varies among the tests.

Evaluation of these experiments is extremely difficult because they involve hierarchies with large differences in cluster sizes. Ideally, the same search strategy is used throughout, the amount of search effort is constant, and the amount of hierarchy degeneration is observed in the precision-recall curves. With small deviations, these conditions are met in the previous tests because comparisons are made within the same hierarchy. In the present tests, both the strategy and search effort cannot be held constant even though the file is always assumed to be freshly re-sequenced so that maximum search speed is attained (no items reside in overflow areas). The difficulty in observing the degeneration arises from the following conditions:

a) the larger the percentage of updating, the larger the final cluster sizes;

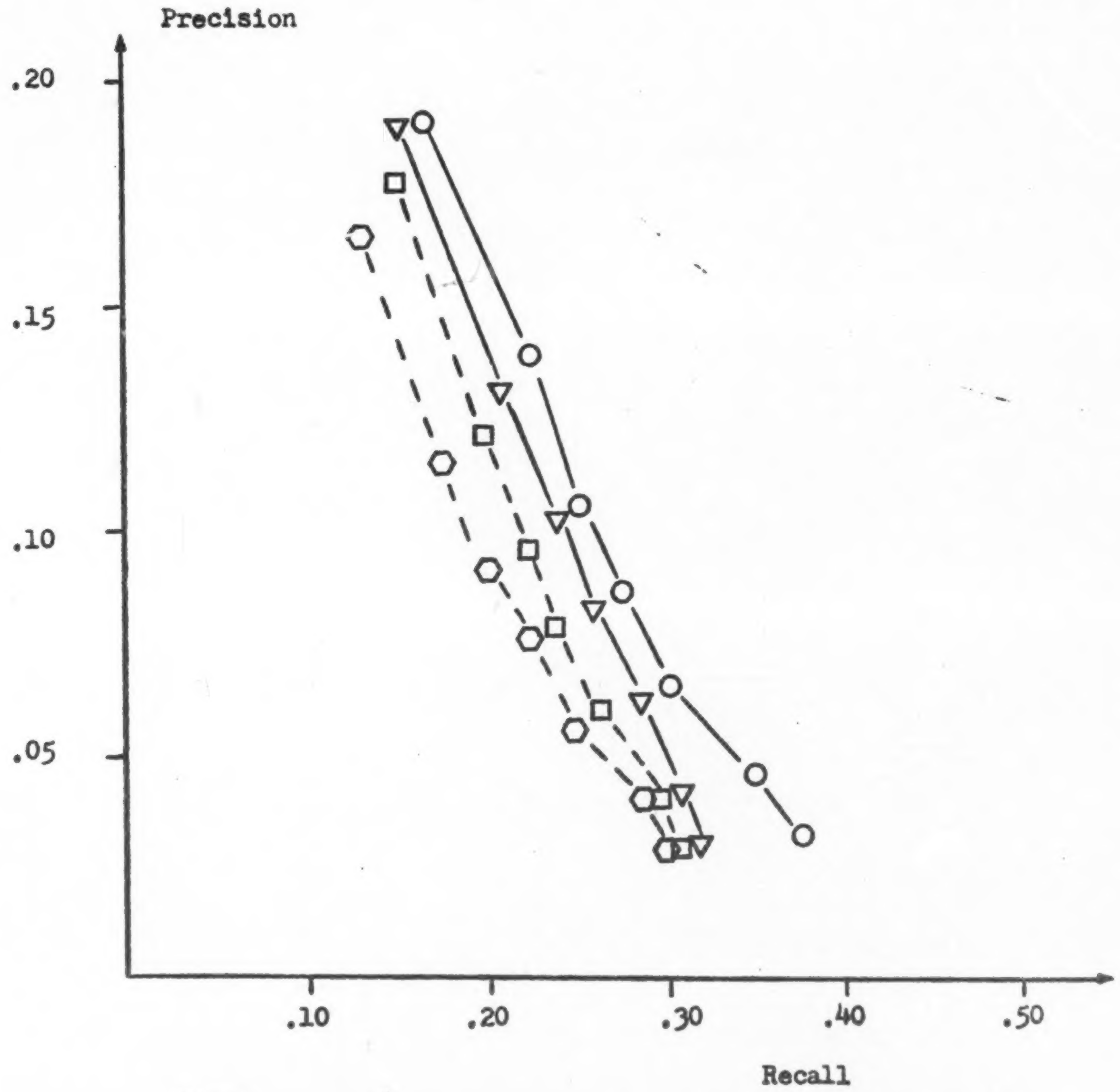
b) since the same number of clusters are expanded in all tests, differences in cluster sizes among the hierarchies makes it impossible to keep the number of document correlations relatively constant in all cases; and

c) both precision and recall generally increase as the number of document correlations increase.

The circular nature of these conditions implies that a performance improvement might be observed with increased amount of updating simply because more documents are examined. This is somewhat true if a constant search strategy is maintained since only large amounts of hierarchy degeneration would show in P-R curves. Consequently, there are two schemes for observing the desired effect: (1) to maintain a constant search strategy and judge degeneration from the P-R differences and the number of profile and document correlations and (2) to alter the search strategy among runs to equalize the number of document comparisons before judging degeneration from P-R differences. The second method equalizes the number of comparisons, and consequently does not measure degeneration using exactly the same search procedure in all cases.

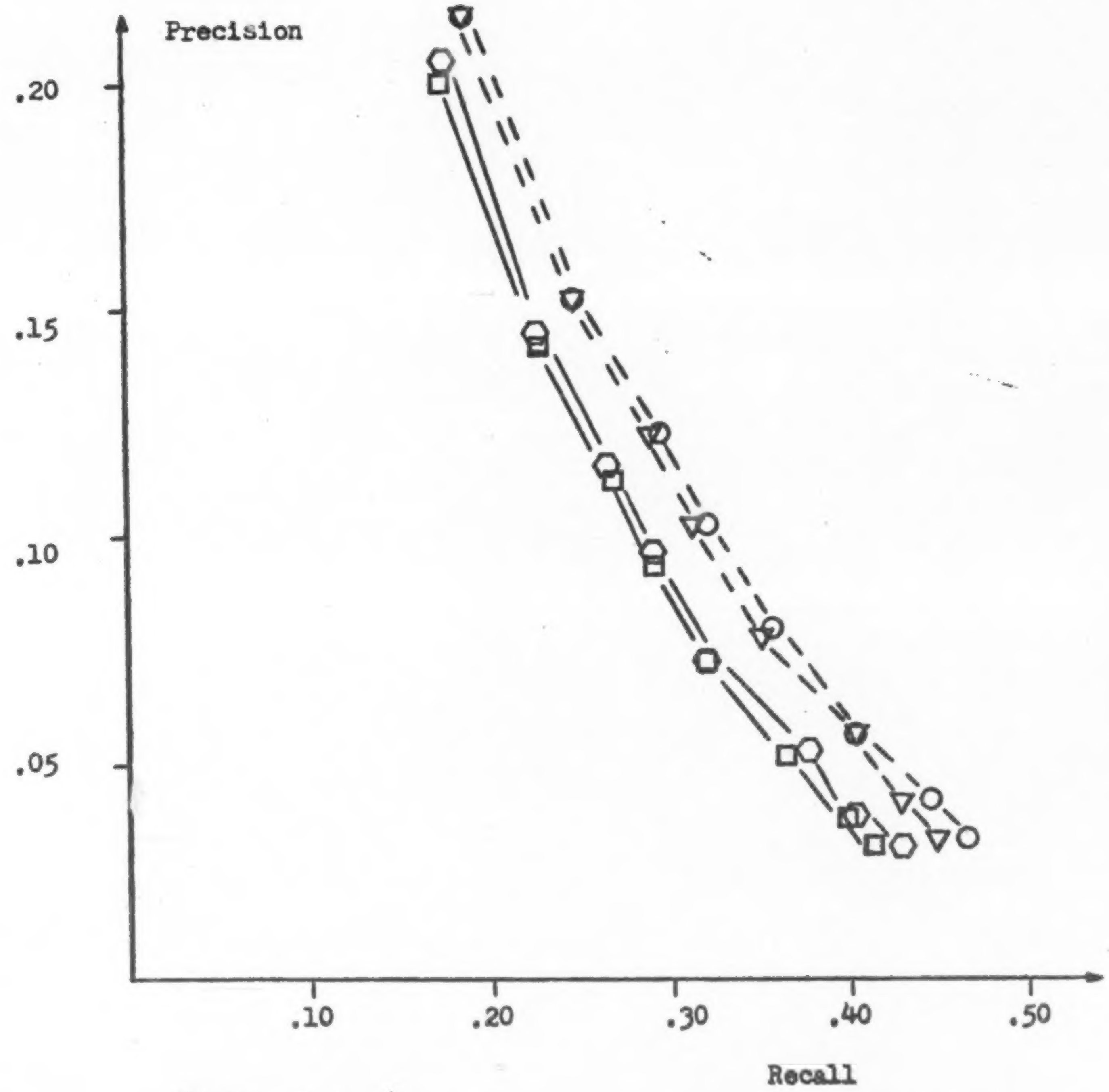
In the following experiments, various fractions of the Cranfield collection are used in updating in conjunction with each of the three maintenance schemes for weighted profiles (NEW, ALTER, NONE). The original profiles, before updating, are of the $P_3^*(\delta = -1)$ type; afterwards the vectors are either of the same type (NEW, NONE) or hybrids (ALTER). For each maintenance scheme two complete searches are made using the narrow and broad SMART search strategies outlined in Chapter IV. The resulting P-R curves allow evaluation by method 1. For example, Figures VI-6, 8, 10 show the P-R plots, normalized measures, and number of correlations per level, C(x), for four narrow searches on document collections subjected to various amounts of updating. In all tests, the same search strategy is used. Figures VI-7, 9, 11 show similar data for a broad search. Especially for the narrow strategy, performance drops off steadily as the amount of updating increases, even though the latter searches are helped in that they examine more documents (120-150). In order to employ evaluation method 2, the same number of document correlations must be performed in all cases. Fortunately the narrow searches for 50% and 75% updating and the broad searches for 0% and 25% updating examine about the same number of documents so it is possible to observe the performance loss from updating in this way also. (The applicable curves are joined by dashed lines in pairs of figures: V-6 and 7, V-8 and 9, V-10 and 11.) Both evaluation schemes suggest there is degeneration of the hierarchy, but their estimates differ moderately.

The precision-recall curves for the complete set of searches under all three profiles maintenance options are contained in Figures VI-6 to VI-11. As always, a number of comparisons and observations can be made. First, consider the curves for 0% and 25% updating. In all broad searches the number of document correlations is approximately the same, and there is virtually no performance difference with this percentage of file growth. For narrow searches, about 20 more document correlations are involved in



Symbol	Hier-archy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.627	.370	13	8.0	86
○	4	25	.650	.399	9	5.2	103
□	5	50	.621	.360	6	4.7	120
⬡	6	75	.613	.345	3	4.7	150

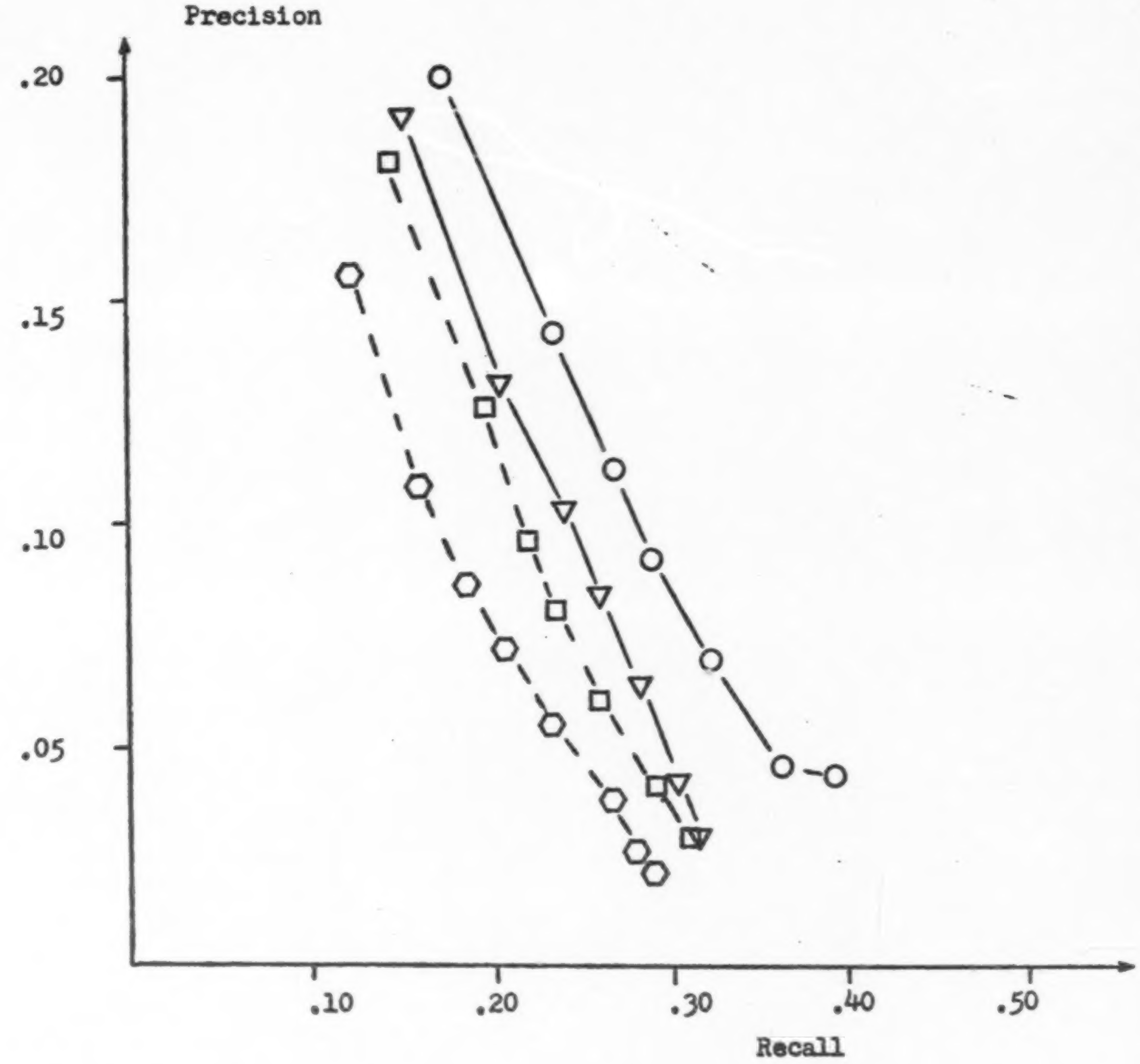
Hierarchy Degeneration Resulting from Updating
Narrow Search, Maintenance Procedure = NEW
Figure VI-6



Symbol	Hierarchy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.695	.440	13	13	162
○	4	25	.696	.449	9	7.9	166
□	5	50	.673	.417	6	6.9	182
⬡	6	75	.671	.419	3	6.6	247

Hierarchy Degeneration Resulting from Updating
Broad Search, Maintenance Procedure = NEW

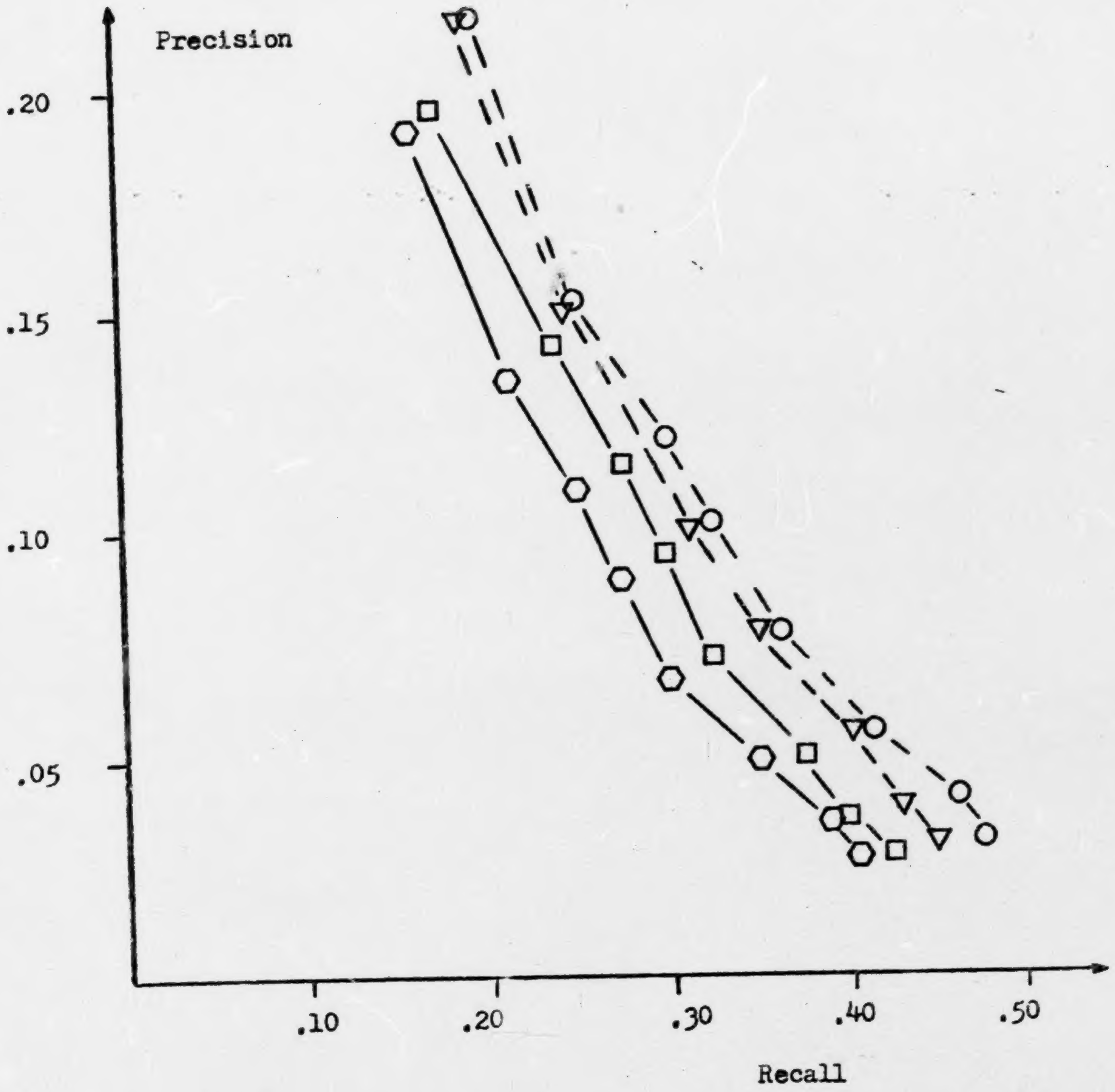
Figure VI-7



Symbol	Hier-archy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.627	.370	13	8.0	86
○	4	25	.664	.413	9	5.7	107
□	5	50	.620	.358	6	4.5	118
⬡	6	75	.593	.326	3	4.4	143

Hierarchy Degeneration Resulting from Updating
Narrow Search, Maintenance Procedure = ALTER

Figure VI-8

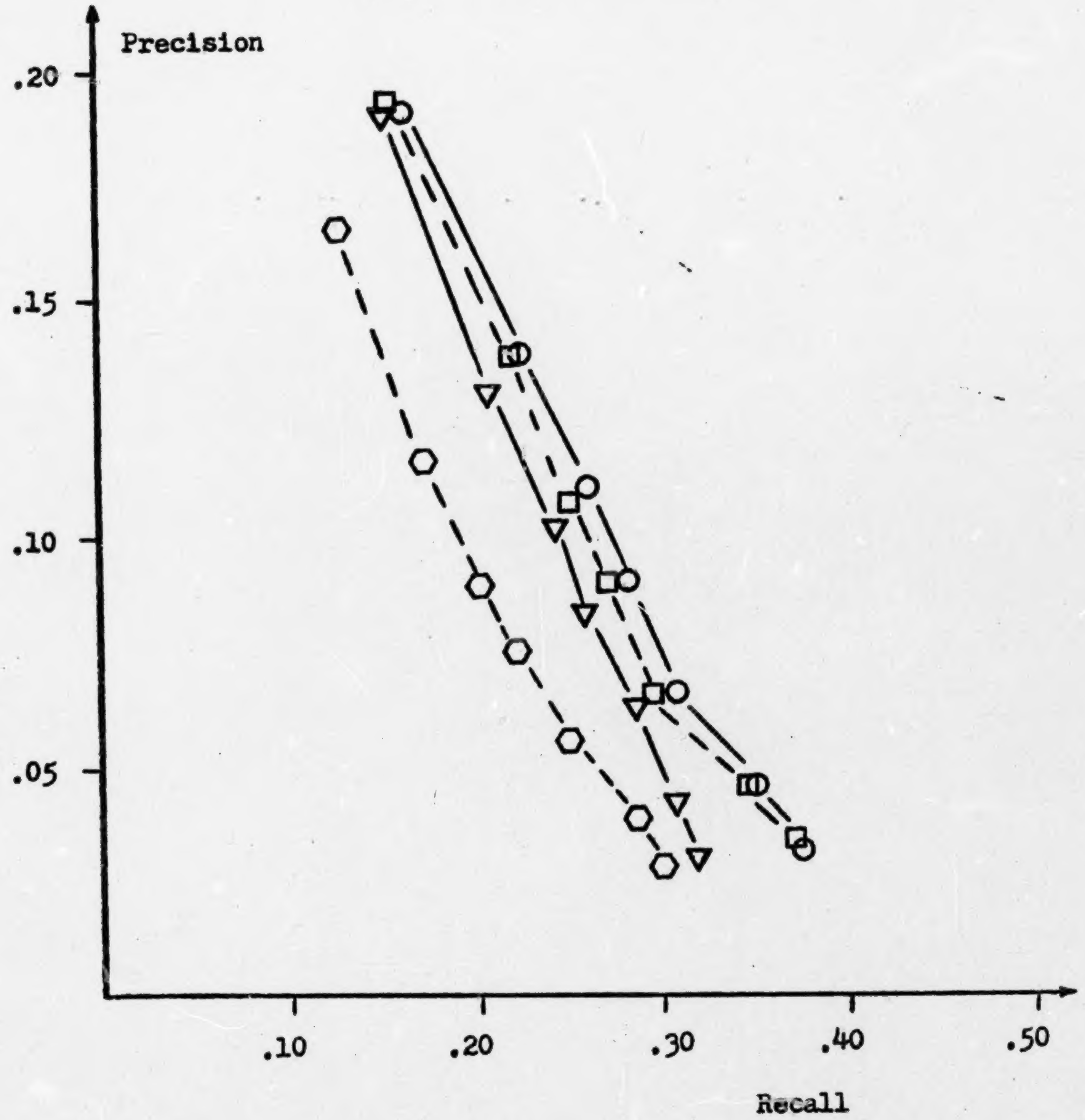


Symbol	Hier-archy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.695	.440	13	13	162
○	4	25	.706	.457	9	8.1	167
□	5	50	.676	.423	6	6.9	177
⬡	6	75	.652	.398	3	7.6	241

Hierarchy Degeneration Resulting from Updating

Broad Search, Maintenance Procedure = ALTER

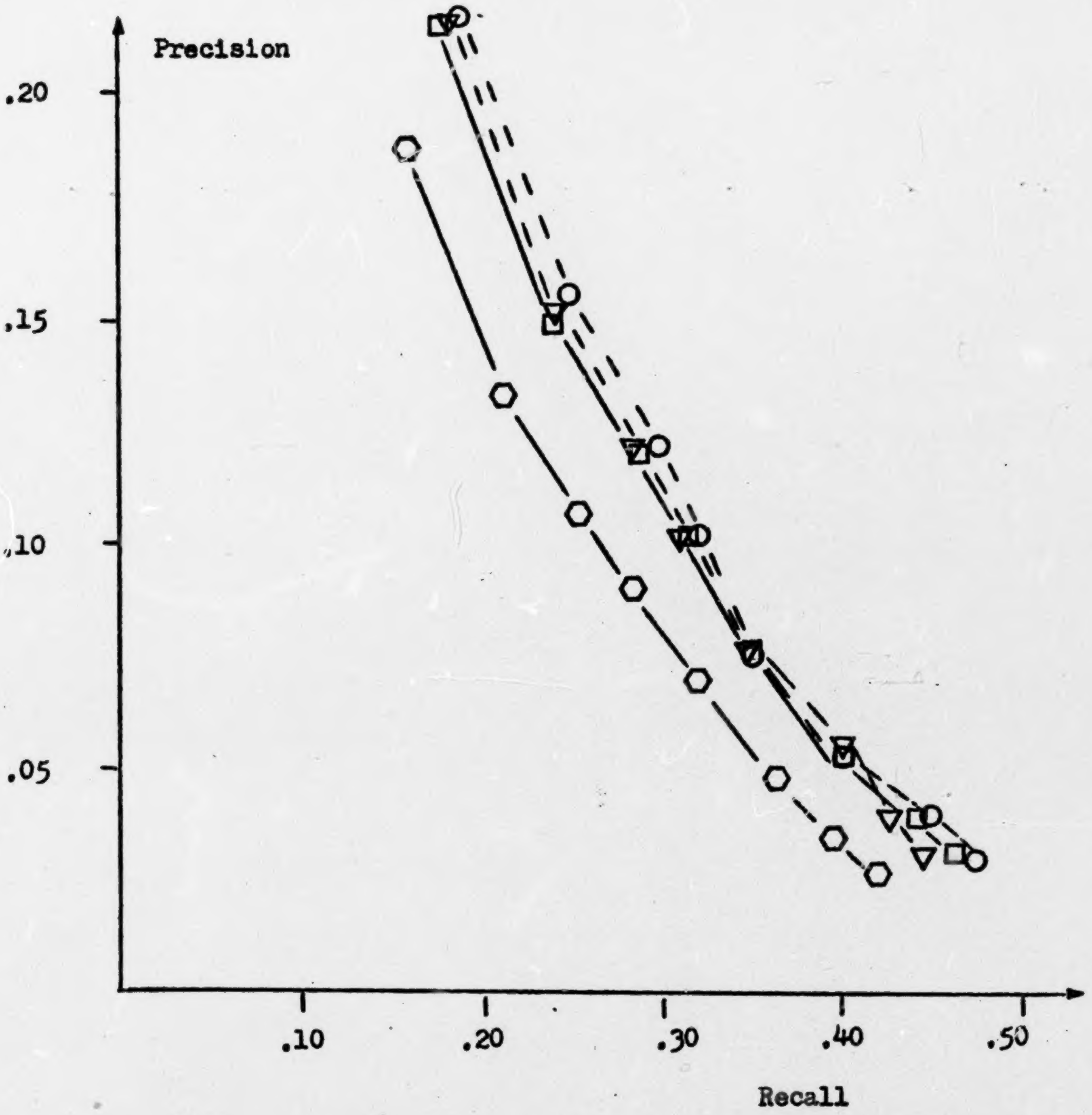
Figure VI-9



Symbol	Hier-archy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.627	.370	13	8.0	86
○	4	25	.656	.402	9	5.8	103
□	5	50	.657	.397	6	5.4	159
⬡	6	75	.618	.348	3	4.7	142

Hierarchy Degeneration Resulting from Updating
Narrow Search, Maintenance Procedure = NONE

Figure VI-10



Symbol	Hier-archy	% Update	NR	NP	C(1)	C(2)	C(3)
▽	1	0	.695	.440	13	13	162
○	4	25	.700	.451	9	8.4	167
□	5	50	.697	.445	6	7.5	247
⬡	6	75	.667	.412	3	8.2	241

Hierarchy Degeneration Resulting from Updating
Broad Search, Maintenance Procedure = NONE

Figure VI-11

the 25% update searches and their P-R curves are better, as expected. Still, the collection seems to accommodate at least 25% updating with little or no performance loss. Second, in nearly all cases the curves for 50% and 75% updating lie substantially below the search simulating a freshly re-clustered collection (0% update). The P-R differences are generally smaller in the broader searches, but the corresponding increase in the number of document correlations to achieve that performance assures that significant degeneration has occurred with this many new additions. This observation is substantiated further by comparing performance between the paired narrow and broad searches mentioned earlier (evaluation method 2). Roughly speaking, the normalized measures drop about 4% for 50% updating and 8% for 75% updating, and the P-R curves remain far apart. Third, all results are somewhat invariant with the profile maintenance procedure. This emphasizes the findings in the previous section; namely that all profile maintenance procedures are roughly equivalent with some preference, perhaps, for hybrid vectors.

The conclusion of these experiments is that enough hierarchy decay occurs with 25%-50% updating to warrant re-clustering. The "break even" point is probably on the low side of this range. The implications for partial re-clustering are obvious. To review, under this procedure each profile includes the number of documents in its current crown and the number of additions since the latest classification (update count). Consequently whenever the ratio of update count to crown size exceeds .25-.50, then all items beneath the node are re-clustered. An alternative to this scheme is to re-cluster all documents in the node's filial set as well.

5. Summary

This chapter considers the problem of updating a clustered file and two very important questions: (1) how profiles should be altered to more accurately represent both new and old documents and (2) how frequently the collection (or node) must be re-clustered in order to recover from the performance loss inherent in the classification-update process. The results from both sets of experiments contain the following information.

a) Weighted profiles have updating characteristics superior to those of unweighted profiles in that they result in fewer relevant clusters which are expanded earlier in searching.

b) A clustered collection can tolerate 25%-50% updating before partial or total re-clustering is necessary.

c) All profile maintenance procedures tested (NEW, ALTER, NONE) performed about equivalently. In particular, new terms do not need to be added to profiles even if the original vectors have undergone earlier term deletion. The recommended scheme is that of forming hybrid vectors as a result of updating (ALTER option). This scheme allows changes to weights of existing profile terms, but keeps the original profile length so the updated vector may overwrite its predecessor.

These conclusions are based on test procedures which use a deliberate partition of the collection and somewhat unique ways of comparing precision-recall curves. It is hoped that neither of these techniques unfairly bias the results.

References

1. V. Lum, H. Ling, M. Senko, Analysis of a Complex Data Management Method of Simulation Modeling, Information Services Department, IBM Research Laboratory, San Jose, California, 1970.

2. A Description of AMIGOS, Comress Corporation, Rockville, Maryland, 1970.

3. M. Kerchner, Dynamic Document Processing in Clustered Collections, Doctoral Thesis, Scientific Report No. ISR-19, Department of Computer Science, Cornell University, October 1971.

Chapter VII

Experiments with Hierarchy Storage

1. Introduction

As stated in Chapter II, the physical file organization of interest is the indexed sequential access method (ISAM) as opposed to the logical file organization which, of course, is based on clustering. The experiments in this chapter have a dual purpose related to physical organization: (1) to determine the I/O delays while searching a clustered file and (2) to compare the appropriateness of storing profiles in level and heir-filial order. I/O delays are measured by the number of disk accesses for obtaining index information and actual data items. The previous experiments measure search effort by the number of expanded clusters or correlations per hierarchy level. This is acceptable since the purpose of the tests is to maximize performance (PF-RC, P-R, etc.) for a given amount of work rather than to study search effort itself. Furthermore, specifying the parameter settings for actual file storage would have made previous conclusions less general than desired. This chapter considers the problem of minimizing search I/O time while maintaining a given performance level. The experimental procedure is to determine disk locations for all documents and profiles using the storage algorithm in Section III.6 and to simulate query processing while monitoring track and cylinder positions. Changing the order of item storage allows comparison of the level and heir-filial sequences. Processing is handled so that the searches resemble those made earlier with SMART using Hierarchy 1 and $P_3^*(\delta = -1)$ profiles. Consequently, the precision-recall performance of the earlier tests can be associated with the amount of disk I/O determined here.

2. Procedure

The following experiments are designed to monitor the I/O activity while processing queries through a clustered file. Actual disk storage and access is not involved. Instead, a disk is simulated by constructing a storage map of "cylinder and track locations" for data records, ISAM indexes, and overflow areas. Data from previous searches and other sources is used to construct the map and then employ it in such a way as to accurately approximate an actual disk search. The general organization of an ISAM file is outlined in Section II.3. This section sets forth the specific parameters used in the simulation and in the record storage algorithm as well as the chosen evaluation measures. It also discusses a number of difficult problems associated with physical file organization.

Disk space management is simulated according to the ISAM philosophy using the parameters and data characteristics in Table VII-1. Eighteen tracks of each cylinder are allocated to profile, document, and index storage while the remaining tracks are reserved for future documents (overflow areas). Each cylinder of data is preceded by a track index and the first record of the entire file is the cylinder index (each index size is 300 bytes). The profile sizes are those of $P_3^*(\delta = -1)$ vectors for Hierarchy 1. In the SMART implementation, each index term is stored as a weighted concept number (term identifier) which requires 4 bytes of storage; other storage requirements per document total 96 bytes for header information, citation, etc. The same memory requirements are used here, resulting in the average record sizes shown in the table.

In making the map of record locations on the simulated disk, each track is treated as one physical record and data items are packed onto it using the storage algorithm in Section III.6. This algorithm attempts to balance the amount of wasted disk space and the frequency of storing filial records on more tracks than necessary (thereby requiring extra accesses during retrieval). In cases where it makes a difference, space is traded for access time only if the amount of waste is beneath a chosen threshold (θ bytes per track). Consequently, in actual tests, the pertinent evaluation data is the percent of wasted data space and the percent of filial record sets requiring extra accesses. Table VII-2 shows the results of applying this storage procedure to the profiles for Hierarchy 1. Data is given for the case of sequencing items by level as well as in heir-filial order, although there is little difference in the outcomes. In both cases, ISAM indexes are stored in their appropriate locations and the threshold for waste is 10% of the track size ($\theta = 720$). Therefore when less than 720 bytes remain on the current track, the next filial set of records begins on a new track. This is a rather infrequent situation here since the size of a typical filial set is large relative to θ. In actuality less than 1% of the file space is wasted even though up to 10% is allowed. About 30% of the filial sets are split across an unnecessarily large number of tracks thereby requiring extra accesses in retrieval. Whereas this percentage depends heavily on θ, simulations show it is practically independent of the distribution of filial set size, once this size exceeds track capacity. Overall, the document collection and its 68 profiles require 69 tracks of which 65 are allocated to documents.

Number of disk tracks/cylinder	20
Tracks/cylinder for data and ISAM indexes	18
Tracks/cylinder for overflow (10%)	2
Track size	7294 bytes
ISAM index size (track or cylinder)	300 bytes
Average record size	
level 1 - profiles (141 terms)	660 bytes
level 2 - profiles (70 terms)	376 bytes
level 3 - documents (54 terms)	312 bytes

Parameters Related to Management of Disk Storage

Table VII-1

Property		Store Items by Level	Heir-Filial Order
Threshold for wasted space	(bytes)	720	720
Level 1, profiles (13)			
Size of average filial set	(bytes)	8580	8580
Total storage	(tracks)	1	1
Level 2, profiles (55)			
Size of average filial set	(bytes)	1616	1616
Total Storage	(tracks)	3	3
Level 3, documents (1500)			
Size of average filial set	(bytes)	8480	8480
Total Storage	(tracks)	65	65
Percent of disk space wasted		0.72%	0.60%
Percent of filial sets requiring an extra access (100% = 69)		30%	35%

Hierarchy Storage in Level and Heir-filial Order

Table VII-2

File management requires considering record retrieval as well as storage. For a clustered file accessed through ISAM, most of the retrieval options concern the handling of indexes. The conventions adopted here retain the cylinder index in core memory, once it is accessed, as well as the track index for the current cylinder (only). Each query is processed independently of others in the following manner. Initially the disk arm (set of read/write heads) is positioned just outside the clustered file, as if serving another program. The search begins by obtaining the cylinder index, first track index, and all level 1 profiles. Thereafter, profiles and documents are accessed as nodes are expanded; new indices are fetched as cylinder boundaries are crossed. Only one optimization technique is employed. Because several nodes may be expanded simultaneously, it is possible to know, in advance, the next few desired records. When this is the case, it is assumed that records are obtained in a single sweep across the disk surface rather than by jumping forward and backward.

Several measures of I/O activity are used, all being related to track and cylinder changes; the term access refers to either type of change. Actual timings or conversion of accesses to milliseconds are not made, since they can be misleading. For example, real time I/O delays are a function of the traffic volume in a computing system and are therefore quite variable, particularly for the IBM 2314 (1). Moreover delays vary with file size since a track change in a small file may correspond to a cylinder change in a big file. In this research the average number of disk accesses per query search is used as a measure of I/O activity, being *reasonably accurate and easily converted to milliseconds in specific computing environments. The averages are subdivided into the number of track and cylinder changes per hierarchy level and the number of tracks and cylinders traversed in each change (step size). This breakdown helps relate the results to files of different sizes or located on different storage devices.*

Search strategy has a major influence on the amount of I/O and consequently on the optimum storage sequence for a clustered file. Here, data from previous cluster-oriented evaluation runs are used to simulate actual searches. First, all level 1 nodes are "accessed" from the simulated disk. Second, the top ranked nodes--as identified in the cluster-oriented evaluation--are expanded by accessing their sons. Finally, the level 2 nodes are expanded in the order specified by their cluster-oriented evaluation ranking and the appropriate documents are accessed. Throughout this discussion, *access* implies fetching an item stored on the simulated disk while monitoring the track and cylinder changes. The number of expanded nodes is controlled to approximate the SMART narrow and broad searches used earlier. Thus, it is possible to approximate the I/O delays connected with the P-R curves for these searches (Figures V-38 and V-39, $P_3^*(\delta = -1)$ profiles). Tables VII-3 and VII-4 show how well the expansion data for these tests (simulated searches) matches the actual SMART searches. The small discrepancies are due to the fact that SMART has complex expansion criteria (see Table IV-2) whereas the simulated search uses only cluster size.

The extent to which the 225 Cranfield requests evenly cover Hierarchy 1 is unknown. That is, some clusters may be examined a disproportionate

Property	Search Strategy	
	Narrow	Broad
Level 1		
Number of profile correlations	13	13
Number of expanded nodes	1.7	3.1
Level 2		
Number of profile correlations	8.0	13.0
Number of expanded nodes	3.0	5.3
Level 3		
Number of document correlations	86	162

Average Expansion Characteristics of SMART Searches

Hierarchy 1, $P_3^*(\delta = -1)$ Profiles

Table VII-3

Property	Search Strategy	
	Narrow	Broad
Level 1		
Number of profile correlations	13	13
Number of expanded nodes	1.8	3.0
Level 2		
Number of profile correlations	7.9	13.9
Number of expanded nodes	2.9	5.6
Level 3		
Number of document correlations	85	167

Average Expansion Characteristics of Simulated Searches

Using the Cranfield Query Set

Table VII-4

number of times. To alleviate deviations from this source, a "simulated query set" is also used in these tests, although no queries are actually involved. Instead, for 598 trials, random nodes are accessed on each level while track and cylinder changes are recorded. To be realistic, all level 1 profiles are accessed and randomly selected nodes are expanded. Thereafter, selections are limited to the sons of previous nodes. Comparing the data in Tables VII-3 and VIII-5 verifies that the simulated queries behave like the actual Cranfield requests in that the same number of nodes are expanded in both searches. Corroborated results from both query sets gives additional confidence to the findings of these investigations.

To summarize, the following experiments try to relate I/O activity to P-R performance levels and to select an optimal storage sequence for a clustered file. The test procedure depends heavily on the accurate simulation of the indexed sequential access method. Briefly, the following steps are involved.

a) Select parameters for record sizes and management of disk space (Table VII-1).

b) Prepare the disk map using the proposed storage algorithm and either level or heir-filial item sequence (Table VII-2).

c) Using either real or simulated requests and the expansion parameters of previous searches (Table VII-3), process each request while monitoring its simulated I/O activity. Processing includes obtaining cylinder and track indexes, profiles on level 1, and the sons of all expanded nodes

Property	Search Strategy Narrow	 Broad
Level 1		
Number of profile correlations	13	13
Number of expanded nodes	1.8	3.0
Level 2		
Number of profile correlations	8.0	13.1
Number of expanded nodes	3.1	5.9
Level 3		
Number of document correlations	85	164

Average Expansion Characteristics of Simulated Searches

Using Random Selection of Expanded Nodes (Simulated Queries)

Table VII-5

thereafter. The primary evaluation measures include a breakdown of the number of accesses per hierarchy level (track and cylinder changes) and the average step size between these changes.

The resulting data is used to evaluate file storage and retrieval options in the forthcoming sections.

3. Test Results

The previous section outlines the simulation and evaluation procedures for studying the I/O activity connected with searches of a clustered file. The results for Hierarchy 1 and $P_3^*(\delta = -1)$ profiles are shown in Tables VII-6 and VII-7. Regarding terminology, the total number of accesses is the average number of track or cylinder changes per query for obtaining index, profile, and document data. This is separated into track and cylinder changes per level; the amount of data moved over between accesses (average step size) is included also. For example, a cylinder step size of 1.5 implies that an average of 1.5 cylinders is passed over each time a change is made. An analogous statement applies to track step size, but is obviously bounded by the number of data tracks per cylinder (18).

By nearly every measure, storing the hierarchy in order by levels appears more economical for forward search strategies. The difference is 1-2 accesses for the narrow search and 3-4 accesses for the broad search. The largest contribution to these differences comes from the nodes on level 2, both in the number of track and cylinder changes and in their step sizes. To review, both storage sequences keep filial record sets

Property	Search Run			
	A	B	C	D
Query Set	Real	Simulated	Real	Simulated
Storage Sequence	Level	Level	Heir-filial	Heir-filial
Total Number of Accesses				
Level 1	2.00	2.00	2.00	2.00
Level 2	1.39	1.42	2.82	3.06
Level 3	6.34	6.62	6.24	6.73
Total	9.73	10.0	11.0	11.8
Number of Cylinder Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	0	0	1.23	1.18
Level 3	1.24	1.22	1.14	1.36
Step Size of Cylinder Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	0	0	1.35	1.48
Level 3	1.59	1.59	1.52	1.57
Number of Track Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	1.39	1.42	1.59	1.19
Level 3	5.10	5.41	5.09	5.36
Step Size of Track Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	1.21	1.27	8.50	7.50
Level 3	2.89	3.04	2.64	2.84

I/O Activity in a Simulated Narrow Cluster Search

Table VII-6

Property	Search Run			
	A	B	C	D
Query Set	Real	Simulated	Real	Simulated
Storage Sequence	Level	Level	Heir-filial	Heir-filial
Total Number of Accesses				
Level 1	2.00	2.00	2.00	2.00
Level 2	1.88	1.90	4.36	4.86
Level 3	11.5	11.7	12.0	12.5
Total	15.4	15.6	18.3	19.4
Number of Cylinder Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	0	0	1.66	1.74
Level 3	1.86	1.85	2.14	2.51
Step Size of Cylinder Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	0	0	1.23	1.25
Level 3	1.08	1.29	1.43	1.48
Number of Track Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	1.88	1.90	2.70	3.12
Level 3	9.64	9.85	9.81	10.0
Step Size of Track Changes				
Level 1	1.00	1.00	1.00	1.00
Level 2	1.24	1.12	7.99	7.09
Level 3	2.45	2.62	2.67	2.88

I/O Activity in a Simulated Broad Cluster Search

Table VII-7

together. In addition, order by levels places all nodes on the same level in adjacent disk locations. Heir-filial order keeps parent nodes and their sons somewhat close at the expense of storing structurally unrelated nodes in separated areas. Consequently, heir-filial order should be more economical in forward, narrow searches since all data resides in a localized area. The choice of the better sequence should hinge on the relative frequency of narrow and broad searches in an actual operating environment. However, it is doubtful that a significant proportion of actual searches are narrower than those used here and thus able to take advantage of the economics of heir-filial order. Most searches involve expansions of one or more unrelated nodes and thereby benefit from storing the hierarchy by levels even more than shown here.

If the file were larger than the 1400 Cranfield documents, the conclusions should be roughly the same even though there are more nodes. With order by levels, there would be greater distance between parent and sons (more space for the read heads to travel), while heir-filial order still confines related data to a localized area. On the other hand, if heir-filial order were used, structurally unrelated nodes have even greater separations and jockeying back and forth between them in a search is quite costly. For example, it is likely the large track step sizes in the tables would turn into steps over cylinders. Figuring both storage schemes are penalized equally, the choice of optimal order comes down to the frequency of extremely narrow searches. As mentioned, few searches are assumed to be as narrow as the one used here, so storage order by levels probably remains the better choice even for larger collections.

These conclusions hold for the case of forward search strategies. Backtracking or plunge-first strategies make a better fit with heir-filial order since it localizes structurally related records. In fact, the SMART searches use a small amount of backtracking and their I/O requirements are only approximated by the present forward searches. The actual I/O is a bit more than the figures quoted in the tables, say 2-3 additional accesses.

It is instructive to compare these cluster searches with a full search in terms of their performance (normalized measures) and the number of correlations and accesses. Such a comparison is given Table VII-8, using the full search in Figure IV-5, the cluster searches in Figures V-38 and V-39, and the I/O data in Table VII-6 and VII-7. In general, the comparison shows that cluster searching achieves its primary goal of recovering a good share of the relevant documents at much less cost than a full search. The narrow cluster search results in NR-NP values which are 60%-70% of those for a full search, but does 8%-15% of the work. A broad cluster search achieves 70%-80% of the full search NR-NP performance for 13%-24% of the effort. Clearly the quality of performance increases as the amount of work increases, but with diminishing returns. Assuming that I/O delays are the dominant factor in determining response time and that one access is made per second, then the cluster searches should finish in 10-16 seconds while the full search requires over a minute. In any case, the actual computer cost and time delay should be predicted rather accurately by the combination of the number of correlations (CPU computation) and disk accesses (I/O and system overhead).

Property	Narrow Cluster Search	Broad Cluster Search	Full Search
Normalized Recall	.63	.70	.88
Normalized Precision	.37	.44	.61
Recall Ceiling	.32	.46	1.00
Number of Correlations			
Level 1 - Profiles	13	13	0
Level 2 - Profiles	8	13	0
Level 3 - Documents	86	162	1400
Total	107	188	1400
% of Full Search	8%	13%	100%
Number of Disk Accesses			
Level 1 - Profiles	2.0	2.0	0
Level 2 - Profiles	1.4	1.9	0
Level 3 - Documents	6.3	11.5	65.0
Total	9.7	15.4	65.0
% of Full Search			

Relation of Performance and I/O Activity for

Cluster and Full Searches

Table VII-8

4. Summary

The present chapter describes a series of experiments related to the indexed sequential access method for managing a disk resident clustered file. Their results are as follows.

a) For forward search strategies, storing the hierarchy by successive levels gives the most economical searches under a variety of conditions. This finding is likely to hold for larger collections and search strategies with some backtracking also.

b) Cluster searching can retrieve many relevant documents with much less system effort than a full search. In the test cases, a cluster search using 10-16 accesses achieves about 70% of the performance (NR, NP) of a full search requiring 65 accesses.

A number of other issues related to physical file organization are discussed--handling of ISAM indexes and overflow space, space-time trade-offs in file storage, and evaluation measures for disk I/O.

References

1. J. Abate, H. Dubner, S. Weinberg, Queueing Analysis of the IBM 2314 Disk Storage Facility, JACM, Vol. 15, No. 4, October 1968.

Chapter VIII

Experiments with a Query Alteration Scheme Based on a Cluster Hierarchy

1. Introduction

Chapter III states that a clustered document file is economical only if it reduces search time or provides facilities not available in other organizations. It also outlines several alternate uses of a cluster hierarchy in an attempt to help justify its construction. Particular attention is given to the concept of associating a substitute (closely related term) with each base profile term. The result is a structure which combines the functions of a thesaurus for query expansion and a directory for file searches. The use of thesaurus classes (substitutes) in combination with clustering is new and holds two distinct advantages. First, since each node is equipped with its own set of substitutes, there is a unique opportunity for using term-term associations from a group of highly related documents (those beneath the node). Consequently, local, narrow term relationships are accommodated on lower hierarchy levels and broader, general relationships are handled on upper levels. Second, substitutes can be used in several ways depending on which ones are applied and how they are matched. If term substitutes improve retrieval, then the utility of a clustered file is increased.

This chapter seeks to establish the validity of an automatic query alteration scheme using term substitutes. Consideration is limited to the TIED matching option mentioned in Section III.8. That is, only the substitutes of matching base terms may alter a query-profile correlation. The recall experiments employ substitutes from parent nodes; the precision experiments use substitutes from nodes on the current level. In all cases, term substitutes are generated by the same algorithm, although certain minor changes are allowed. Briefly, for any given profile term (base), the algorithm identifies its substitute as another term in the same profile which has the largest term-term correlation in the documents beneath that node. Consequently, the derivation could involve computing large similarity matrices. Fortunately, the magnitude of this task can be reduced. Specifically, the upper level matrices can be formed from those on lower levels. Furthermore, the previous experiments suggest that only the most important profile terms need be considered and the use of shortened profiles, $P^*_3(\delta = -1)$, in the experiments reduces the effort further. These reductions are important since most work with thesaurus construction and term-term relationships is greatly hampered by computing, storing, and handling large matrices.

In the final analysis, the proposed alteration scheme does not improve retrieval, at least for the options tested. However, it appears useful as part of a search feedback scheme. The concept of augmenting each profile with its own thesaurus (term substitutes) remains intriguing; in view of success in similar experiments with unclustered collections, further work is warranted.

2. Deriving Base-Substitute Pairs

Base substitute pairs are profile terms with maximum term-term correlation among the documents in a node's crown. For a detailed explanation of this concept, consider a hierarchy without overlap such as that in Figure VIII-1a. For any node n with profile P, let T be the term-document matrix representing the documents in its crown. As shown in Figure VIII-1b, the element T_{ij} is the weight of the j^{th} concept in the i^{th} document and a row T_{i*} corresponds to a complete document. Suppose an association matrix $A = T'T$ is formed, where the ' indicates matrix transpose. Then the cosine similarity between the i^{th} and j^{th} terms with respect to node n is

$$S_{ij} = A_{ij} / \sqrt{A_{ii} A_{jj}} \qquad \text{(VIII-1)}$$

Once the matrix S is obtained, the substitute for the base profile term i is another profile term j such that

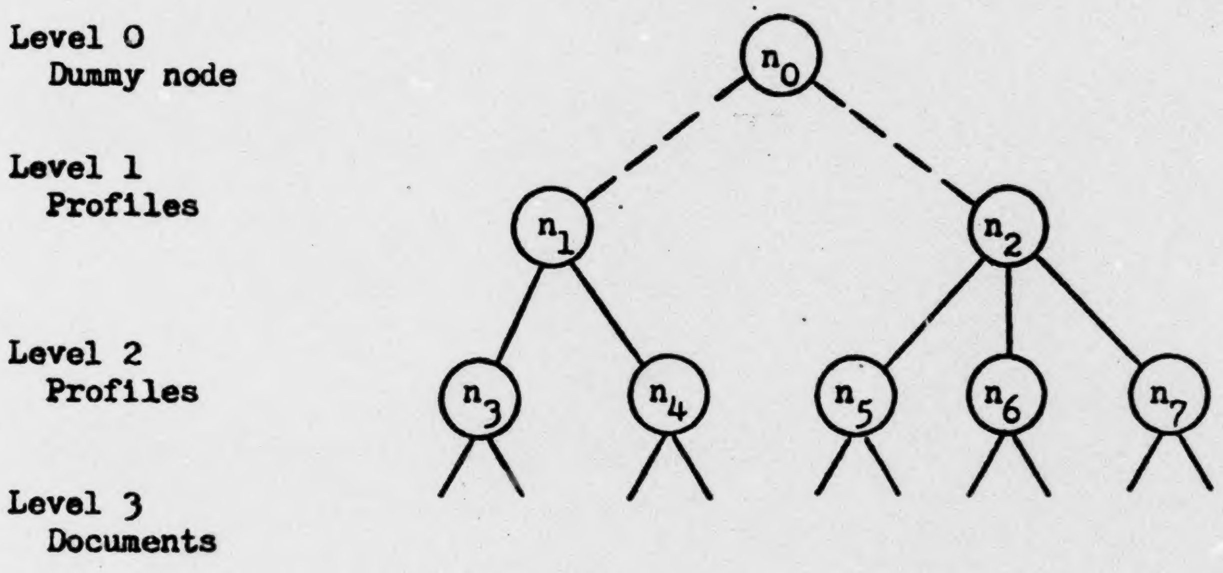
$$S_{ij} = \text{MAX} \left\{ S_{ik} \;\middle|\; k \neq i,\ k = 1, 2, \ldots, v \right\} \qquad \text{(VIII-2)}$$

In practice, there are more terms in T than in P because of term deletion in profile construction. Consequently, the association matrix A can be limited to those terms in P. This saves a considerable number of term-term correlations and does not alter the above computations.

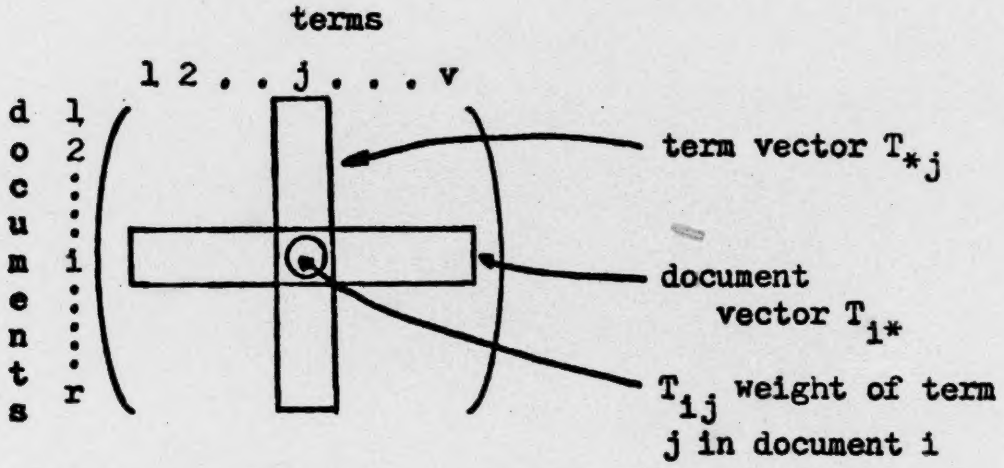
The real problem is finding an economical algorithm for obtaining the similarity matrix S for each node. The difficulty is that T is available only by rows (document vectors) while direct calculation of S requires its columns. Transposing T to make its columns accessible requires a moderate amount of computation; this operation is equivalent to inverting the set of document vectors to get term vectors. Another algorithm for obtaining A is to accumulate its elements as documents are input and to avoid forming T' altogether. The following steps would be executed:

a) initialize $A = 0$;

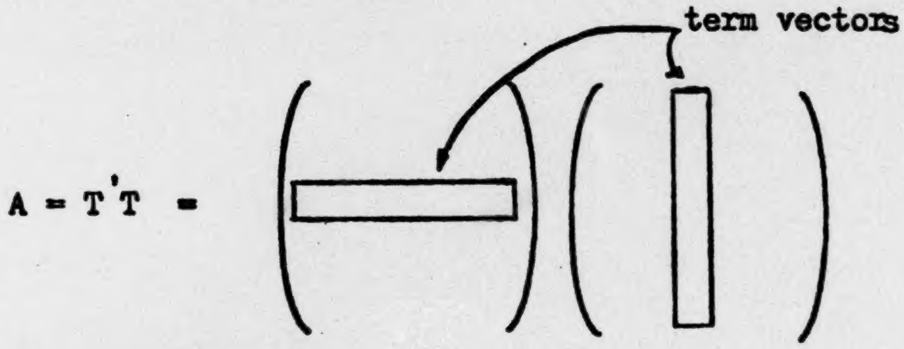
b) read the next document $D = (d_1, d_2, \ldots, d_v)$;



a) Sample Cluster Hierarchy



b) Term-Document Matrix



c) Association Matrix

Figure VIII-1

c) set $A_{ij} = A_{ij} + d_i d_j$ for $i, j = 1, 2, \ldots, v$;

d) repeat steps b) and c) for all documents in the crown of node n.

For a small number of profile terms, either technique is acceptable. For a moderate number of terms, accumulation is preferable since each document is handled only once. However, if the association matrix cannot be made core resident, the inversion technique must be used regardless of other considerations.

The similarity matrices for the lowest level nodes can be obtained economically by one of the above algorithms since only a few documents are involved in the computation. Matrices for upper level nodes could be derived in the same way, but with considerable duplication of effort. The fact is that the association matrices on level i are calculable directly from those on level i + 1. To illustrate, consider node n, and its sons n_2 and n_3 in Figure VIII-1a. Since there is no further need to refer to specific elements of matrices, let T_k, A_k, and S_k denote the document-term, association, and similarity matrices in conjunction with node n_k. Using the assumption that the hierarchy contains no overlap, T_1 can be viewed as partitioned so that

$$\begin{aligned} A_1 &= T_1' T_1 = (T_2' \mid T_3') \left(\frac{T_2}{T_3} \right) \\ &= T_2' T_2 + T_3' T_3 \qquad\qquad \text{(VIII-3)} \\ &= A_2 + A_3 \end{aligned}$$

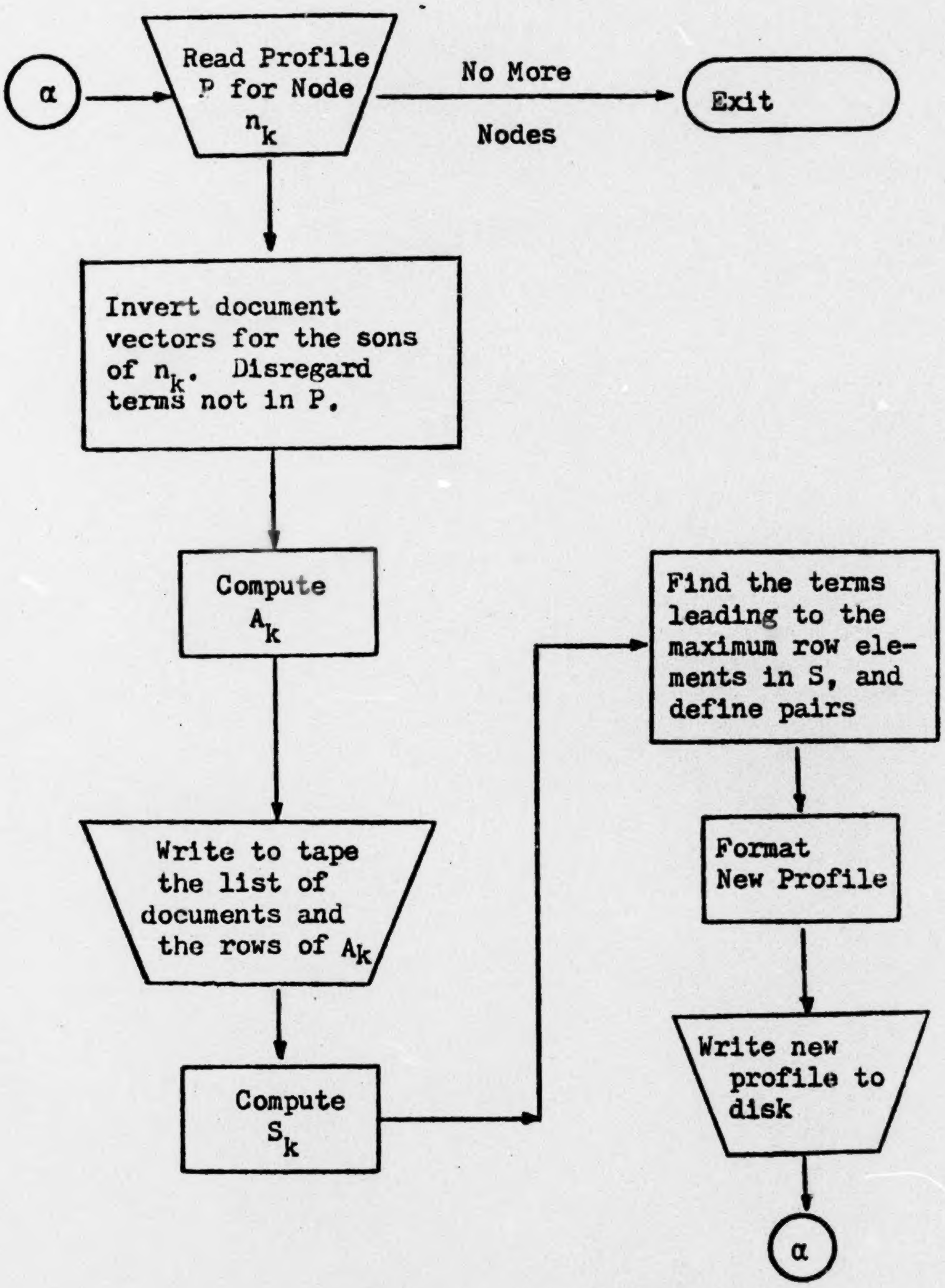
If A_2 and A_3 are saved in magnetic tape files, for example, then A_1 can be obtained by summing corresponding records in each file.

In practice, a clustered collection contains overlap. This can be dealt with also. In the example, let U_2 and U_3 denote the documents unique to nodes n_2 and n_3 respectively, and let X denote their common documents. Then

$$T_2 = \begin{pmatrix} U_2 \\ \hline X \end{pmatrix} \qquad T_3 = \begin{pmatrix} X \\ \hline U_3 \end{pmatrix} \qquad T_1 = \begin{pmatrix} U_2 \\ \hline X \\ \hline U_3 \end{pmatrix}$$

$$\begin{aligned} A_1 &= T_1'T_1 = (U_2' \mid X' \mid U_3') \begin{pmatrix} U_2 \\ \hline X \\ \hline U_3 \end{pmatrix} \qquad \text{(VIII-4)} \\ &= U_2'U_2 + X'X + U_3'U_3 \\ &= A_2 + A_3 - X'X \end{aligned}$$

Thus, the effects of overlap can be removed by subtracting a small correction matrix made from the overlapping documents. It is not necessary to actually compute $X'X$ since corrections can be made in place as described in the accumulation technique above. In the case of a document having membership in several clusters, the correction must be applied each time the overlap occurs. Overlap is easily detected if each tape file (association matrix) is preceded by a list of documents used in its preparation. Merging and checking these lists discloses overlap and allows for corrections. Complete flowcharts for finding base-substitute pairs are shown in Figures VIII-2 and VIII-3. Following the previous example, it is assumed that all documents reside on disk and that extra

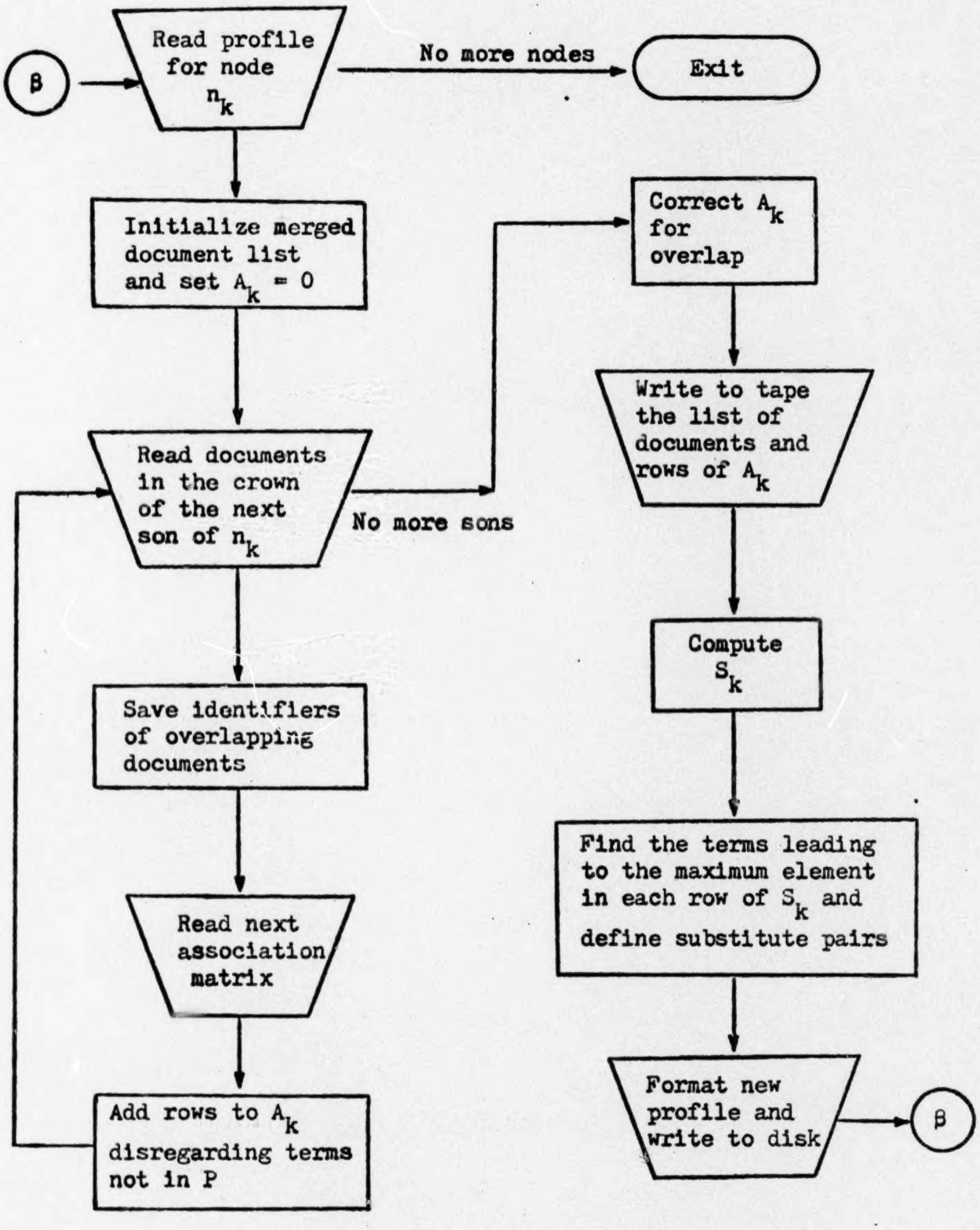

α
Read Profile P for Node n_k
No More Nodes
Exit
Invert document vectors for the sons of n_k. Disregard terms not in P.
Compute A_k
Write to tape the list of documents and the rows of A_k
Compute S_k
Find the terms leading to the maximum row elements in S, and define pairs
Format New Profile
Write new profile to disk
α

Legend: T_k is a term-document matrix

$A_k = T_k' T_k$ is a term-term association matrix

S_k is the matrix of cosine term-term similarities

Figure VIII-2


β
Read profile for node n_k
No more nodes
Exit
Initialize merged document list and set $A_k = 0$
Read documents in the crown of the next son of n_k
No more sons
Save identifiers of overlapping documents
Read next association matrix
Add rows to A_k disregarding terms not in P
Correct A_k for overlap
Write to tape the list of documents and rows of A_k
Compute S_k
Find the terms leading to the maximum element in each row of S_k and define substitute pairs
Format new profile and write to disk
β

Formation of Term Substitutes on Upper Hierarchy Levels

Figure VIII-3

tape or other storage is available as needed.

The discussion above suggests that corrections should be made for the effects of cluster overlap via equation VIII-4. It can be argued that a small amount of overlap could be neglected since it has little effect on the final similarity matrix. However, it may not be desirable to correct for overlap on the grounds that terms in these documents characterize several topics (clusters) and thus should receive additional emphasis when determining the substitute sets for upper level nodes. Generally speaking, neglecting the correction factor $-X'X$ in equation VIII-4 increases similarities S_{ij} when both the i^{th} and j^{th} terms occur in overlapping documents. Consequently, the desired similarity increases are made and the computation of association matrices is simplified at the same time.

This section presents several methods for deriving base-substitute pairs. It is a bit difficult to quantify the savings from using the computation schemes suggested here. However, if a typical shortened profile contains 20% of terms represented in its term-document matrix T then its association matrix contains only 4% of the entries in the full matrix $T'T$. The fact that upper level similarity matrices can be obtained from those on lower levels, clearly saves substantial sort or calculation time regardless of how overlap is handled. _In the following experiments, substitutes are obtained for each term in $P_3^*(\delta = -1)$ profiles for Hierarchy 1._ In all cases, similarity matrices are computed by inverting the documents in a node's crown and finding the cosine correlations directly. Corrections are always made for cluster overlap. Several complete sets of substitutes are generated by applying different frequency and other restrictions to the participating terms. Each set consists of 69 groups of base-substitute pairs--55 for the nodes on level 2, 13 for the nodes on level 1, and one for a dummy node on level 0 as illustrated in Figure VIII-1. Since there is no profile corresponding to the dummy node, the terms used for its base-substitute pairs are the most frequent keywords in the collection, up to 20% of the entire vocabulary.

3. Term Substitutes as Precision and Recall Devices

Previous sections outline various ways of using term substitutes in query searches. When employed as precision devices, substitutes of one node help distinguish it from all others by altering correlations which involve matches on both a base term and its substitute. Presumably it is reasonable to increase the correlation since 1) the base and substitute terms are highly related in the documents beneath the node and 2) both terms are used in the request. The intent, then, is to improve precision by giving more emphasis to combinations of matches on existing query terms rather than by adding new or related terms as in a normal thesaurus expansion. The specific algorithm for modifying a correlation contains the following steps:

a) determine the set of base profile terms B which match the query;

b) using the terms in B, compute an initial cosine correlation C_1 between the query and profile;

c) determine the set $B' \subset B$ whose elements are base terms and their substitutes provided that both are already elements of B;

d) using the terms in B', compute an additional cosine correlation C_2 between the query and profile; and

e) compute the final correlation value $C = C_1 + eC_2$ where e is an experimentally determined emphasis factor.

Clearly, the case of $e = 0$ is the same as not using substitutes; $e > 0$ gives higher correlations to vector matches involving bases and their substitutes; $e < 0$ does just the opposite. Another explanation and an example of this algorithm is given in Section III.8.B. Using the terminology developed there, substitutes are obtained from nodes on the current hierarchy level rather than previous levels and they are TIED to their bases since matches must occur on both a base and its substitute before a correlation is altered.

The precision experiments involve three sets of substitutes for the base terms on level 1 of Hierarchy 1 ($P_3^*(\delta = -1)$ profiles). Each set is generated by the procedure outlined in the previous section, but subjected to one of the following restrictions:

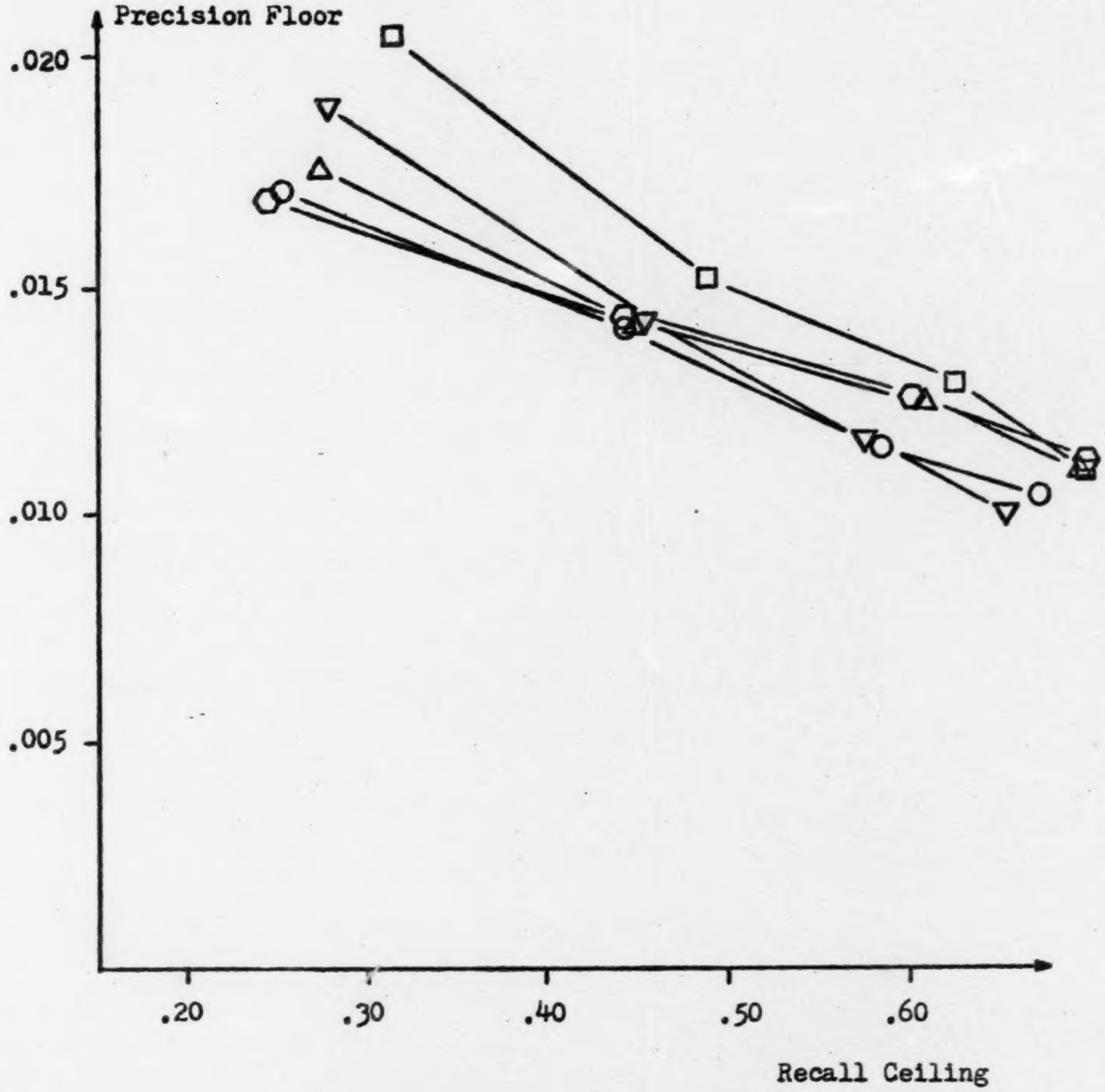
a) no restrictions;

b) only base profile terms of medium frequency are permitted to have substitutes; or

c) the correlation between a base and substitute must lie in the interval (0.45,0.75).

The intent of these restrictions is to alleviate effects from terms of little importance and those having chance relationships with other terms. Figure VIII-4 contains the PF-RC evaluation curves for searches with and without the use of substitutes as precision devices. Unfortunately for the options tested, substitutes do not help discriminate among nodes. Neither frequency nor correlation restrictions nor changes in the emphasis factor (e) have much ability in raising the overall performance to that of a search mode without substitutes. It is not the case that all requests do poorly; a substantial number do slightly better and a few see spectacular improvement. However, the overall trend is on the downward side. For the case of $e = 1$ correlations are probably dominated by matching bases and substitutes since the weight of both terms is effectively doubled (see Chapter V). For $e = -\frac{1}{2}$, the correlation probably depends too much on random term matches. Although $e = 0$ is certainly not an optimal value, these tests indicate there is little to be gained from using substitutes as precision devices.

More frequency, a thesaurus or term substitutes are employed as recall devices, that is, as a source of new keywords for broadening the scope of a request. The additional terms increase the opportunities for query-profile matches and therefore result in higher recall searches. Generally a thesaurus contains term relations appropriate to the entire collection (1). However, with clustered documents, substitutes may be associated with each node (small set of documents) and requests can be altered in a selective careful manner. In the next experiment, the only substitutes influencing a profile correlation are those in the profile's parent nodes; and this set is limited further to those substitutes whose bases have



Symbol	Emphasis Factor (e)	Description of Substitute Set
□	0	None used
▽	$-\frac{1}{2}$	No restrictions
△	1	No restrictions
⬡	1	Medium frequency terms only
○	1	Correlation restrictions (.45,.75)

Relative Merit of Using Substitutes as Precision Devices
$P_3^*(\delta = -1)$ Profiles, Hierarchy 1, level 1

Figure VIII-4

already matched the request. Obviously, nodes on level 1 have no parent; in this case, the substitutes in the hierarchy's dummy node are used as suggested in Section VIII.2. For a precise illustration of the correlation procedure, consider a search which expands only one node on level i. When the request is matched with this node's profile, a set of matching base terms B is obtained; let S be the substitutes for these terms which do not already appear in the query. Each element of S is assigned a weight w and temporarily appended to the request. Using the broadened query, the following matching procedure is applied to each subordinate profile of the expanded node (those on level i + 1):

a) compute an initial profile-query correlation C_1 without using the substitutes in S;

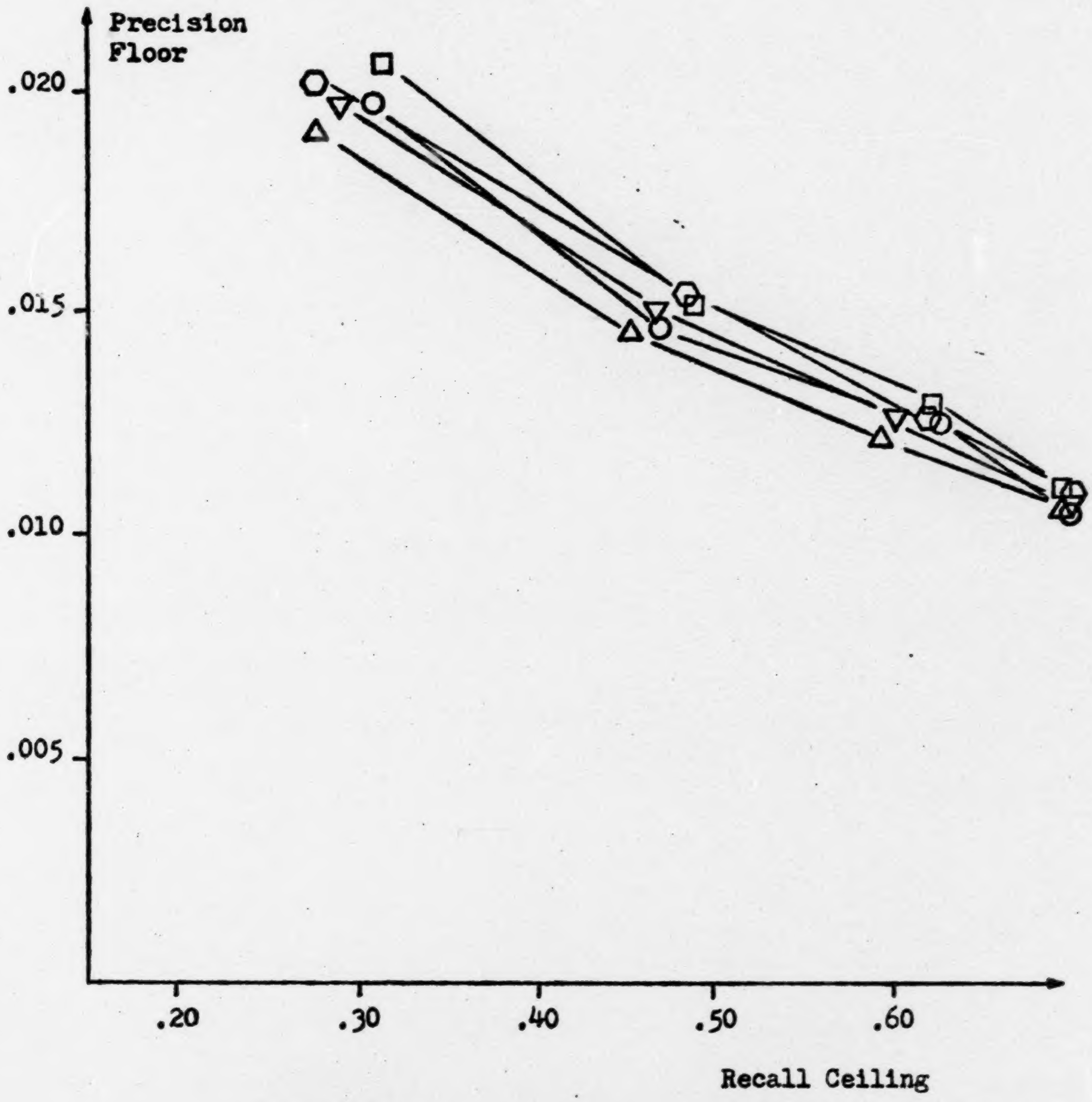
b) determine a set of substitutes $S' \subset S$ whose bases took part in the correlation C_1;

c) compute a secondary correlation value $C_2(w)$ using only the substitutes in S'; and

d) compute the final correlation $C = C_1 + C_2(w)$.

C_2 is clearly a function of w since the substitutes have pre-assigned weights. Clearly w = 0 is the case of not altering correlation values and as w increases so does the importance of the terms used to broaden the request. The above procedure is applicable to searches which expand more than one node; it is a bookkeeping matter to determine which substitutes are to be used. The intent of this seemingly complicated procedure is the careful addition of new request terms from the parent nodes. Furthermore the new terms (substitutes) have an opportunity of affect correlations only when there is a match on the corresponding base (TIED option). An example of this scheme is given in Section III.8.B. One additional remark is required for a complete description. The substitutes added to a request have a temporary existence. That is to say they are replaced by substitutes from expanded nodes on lower levels. Thus the request is altered more and more selectively as the search proceeds and the query vector does not become unnecessarily long.

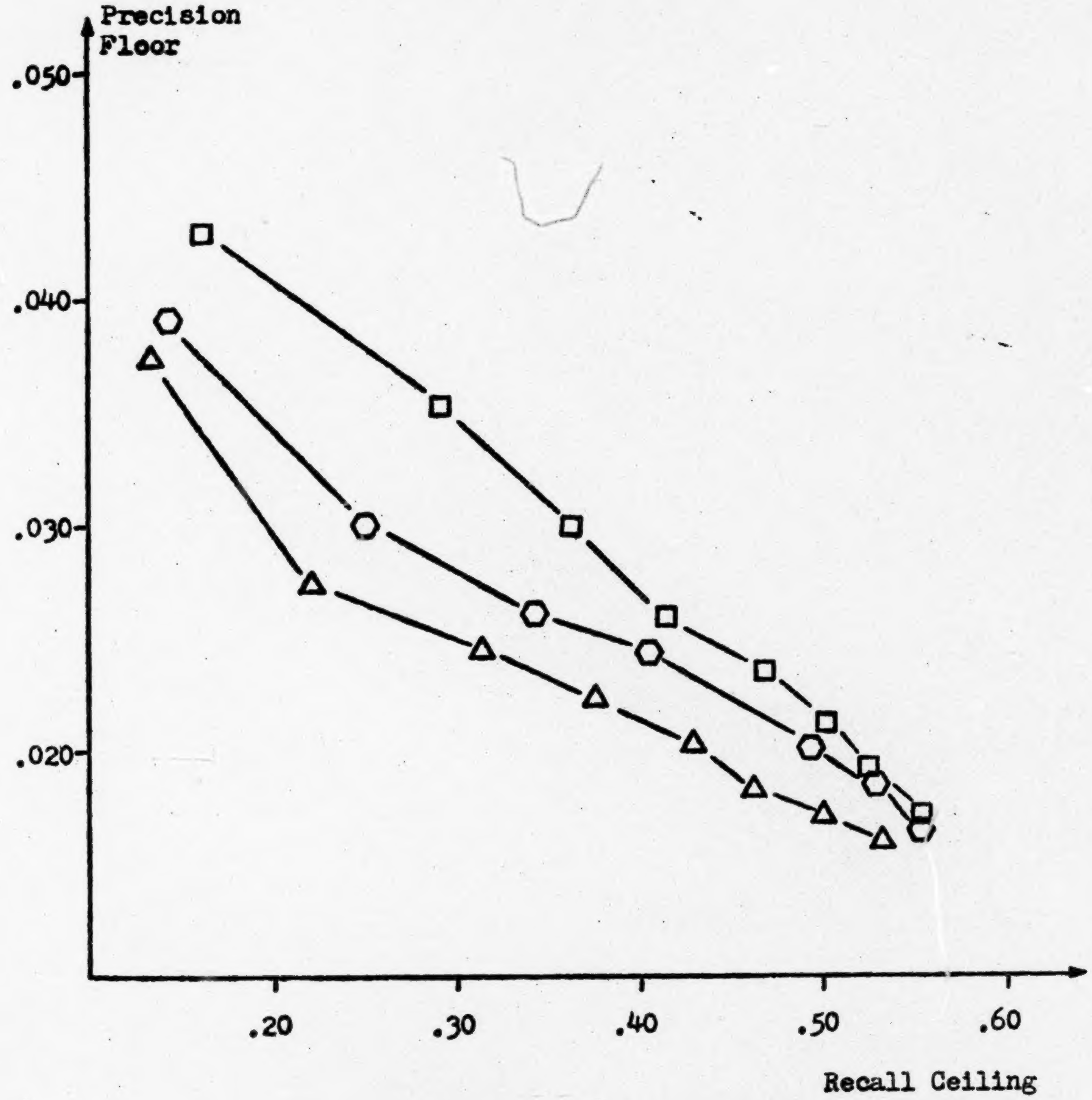
The paragraphs above describe how term substitutes are used as recall devices in these experiments. As before, Hierarchy 1 and $P^*_3(\delta = -1)$ profiles provide the test collection. Two different sets of substitutes are employed, the first of which has no restrictions. In the second set, only medium and high frequency profile terms are permitted to have substitutes and then only if the correlation between each base and substitute exceeds 0.25. Figures VIII-5 and VIII-6 show the PF-RC evaluation curves for searches with and without the use of substitutes as recall devices. In each case, less favorable performance is obtained when terms are added to requests. A smaller performance loss is noted when frequency restrictions are applied, mostly because there are fewer cases in which correlations are altered. Neither changes in the restrictions on the substitute set nor in the weight of terms added to request increase the recall level as desired. Again, it is not the case that all requests do poorly; some show great improvement. Consequently, these techniques could be used as an automatic feedback procedure, particularly when no relevant documents are retrieved in an initial search. For example, the use of substitutes improves the performance of about half the requests



Symbol	Substitute Weight (w)	Description of Substitute Set
□	0	None used
▽	$\frac{1}{2}$	No restrictions
△	1	No restrictions
⬡	1	Medium-high frequency $S_{ij} > .25$
○	2	Medium-high frequency $S_{ij} > .25$

Relative Merit of Using Substitutes as Recall Devices
$P_3^*(\delta = -1)$ Profiles, Hierarchy 1, Level 1

Figure VIII-5



Symbol	Substitute Weight (w)	Description of Substitute Set
□	0	None used
△	1	No restrictions
⬡	1	Medium-high frequency terms $S_{ij} > .25$

Relative Merit of Substitutes Used as Recall Devices
$P_3^*(\delta = -1)$ Profiles, Hierarchy 1, Level 2

Figure VIII-6

for which no relevant are contained in the 2 top-ranked clusters in a normal search.

One problem in both the precision and recall experiments is the modest number of cases in which substitute terms actually influence correlations. The number of cases could be increased, in part, by switching to the UNTIED matching option, that is, giving substitutes the same opportunity for matches as any other term. Another problem is that in many instances, the same base-substitute pair occurs in many nodes and thus gives little discrimination. A partial solution to this may lie in placing different restriction on the terms allowed to have substitutes and the strength of associations among terms. However, it may be that no substitutes can adequately discriminate among profiles because they represent a large number of documents. This might explain why more success occurs when term-term relations are used to distinguish individual documents from each other as in the experiments by Jones (1, 2). Of course, these experiments are not intended to address this broader question.

4. Summary

This chapter describes a query alteration procedure based on sets of term substitutes associated with each node of a cluster hierarchy. The substitute for each base profile term is another profile term which is strongly related to the base in the documents beneath the node under consideration. Depending on how they are applied, base-substitute pairs function as either precision or recall devices. Unfortunately for the options tested here, the use of substitutes result in performance losses rather than gains. Consequently, substitutes should not be included in profiles in their present form. They might be used profitably as part of a feedback procedure or presented to a user browsing through the collection, however. In spite of the results, the basic idea of combining a thesaures and a profile is appealing since both are tailored to the contents of a specific group of documents.

Two significant contributions of this chapter relate to the computation of term-term similarity matrices. It is shown that a complete matrix can be obtained from a set of smaller matrices representing only part of the collection. Furthermore under certain conditions, it is unnecessary to invert the document set to obtain term vectors required in the calculation. Both of these techniques provide reductions in the amount of computer time for generating similarity matrices.

References

1. K. Sparck Jones, E. O. Barber, What Makes an Automatic Classification Effective? Technical Report, University Mathematical Laboratory, Cambridge, England, 1970

2. K. Sparck Jones, Automatic Keyword Classification for Information Retrieval, Butterworths, 1971.

Chapter IX

Comparison of Inverted and Clustered Document Files

1. Introduction

The most widely used file organization in document retrieval systems is the inverted organization. Chapter II discusses the principles of this technique and the space-time tradeoffs involved. The basic idea is to construct a data base which lists a document under entries for each of its index terms. The file is inverted in that it is maintained in term order rather than document order. Actual implementations generally use a combined file approach. That is, documents are stored consecutively on disk, but in such a way as to be individually accessible, for example, through their accession number. In addition, an inverted directory is constructed with one entry per vocabulary term; each entry is a list of accession numbers of documents containing that term. An alternate scheme might place disk addresses in the directory rather than accession numbers. In either case, the search program computes correlations with all documents on directory lists corresponding to query terms. In some instances, including "within document weights" in directory entries allows correlations to be accumulated during the directory scan; therefore, only the highest ranking document citations are ever taken from storage. This is the case for the cosine function; consequently the directory scan is of primary importance in the comparisons made here.

The purpose of this chapter is:

a) to compare the storage requirements for inverted and clustered files,

b) to examine the search speed of an inverted directory as function of query length and collection size, and

c) to compare the speed and effectiveness of inverted and cluster searches.

The test procedure is similar to that in Chapter VII. Briefly, disk storage and retrieval is simulated for a given collection of documents and requests while monitoring simulated I/O activity. Data is tabulated by query length or by hierarchy level in order to make the necessary comparisons. The tests result in the following conclusions.

a) The inverted organization requires twice as much storage space as a clustered file in order to provide equivalent retrieval services. However, if relevance feedback or document space modification are not included in the system, both file organizations require about the same amount of storage.

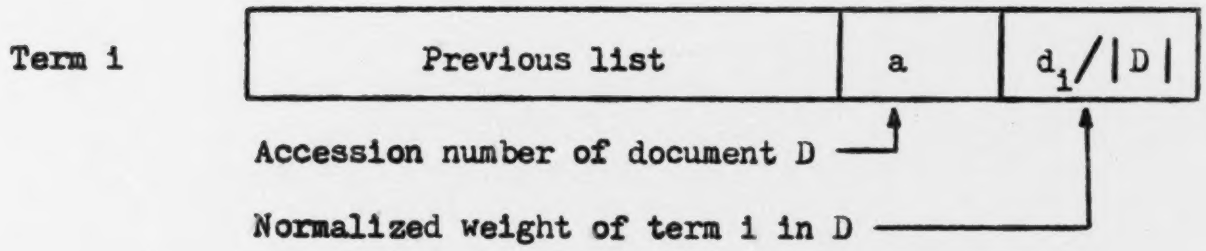
b) Search time (disk accesses) in an inverted file increases with query length (linearly), collection size, and the number of documents retrieved. The directory scan requires 1-2 accesses per term, depending on the collection size. Search time in a clustered file is basically a function of the number of expanded clusters (i.e., search strategy), and somewhat independent of query length and number of retrieved items.

c) For specific number of disk accesses, the inverted file search retrieves a fixed number of documents and achieves high precision at a specific recall level. For the same number of accesses, the cluster search provides the user with many or few documents. Generally the precision is less, but the recall level may be higher or lower depending on the number of retrieved documents.

Naturally, these conclusions are subject to the assumptions, conditions, and search strategies in the various tests. Neither scheme is superior on all points, each one has its own strengths. The least that can be said is that the clustered organization uses no more storage space and provides more flexible searches. That is, searches can be quick and inexpensive, thorough and costly, or aimed at high or low recall. However, they are less precise than inverted searches, in most instances.

2. The Inverted Directory--Storage and Search

In the following experiments the inverted organization is a combination of two files. The first is a consecutive disk file of all documents including citation data, index terms, and weights. The citation is retrieved for user printouts while the terms and weights are required only if relevance feedback or space modification is included in the system capabilities. (See Chapter II for a discussion of this point.) The second part is the inverted directory containing one entry per vocabulary term. Each entry is a list of accession numbers of documents containing a given term and the "normalized term weights" within those documents. For example, a document D with term i of weight d_1 adds the following data to the directory list of that term:


Term i
Previous list
a
$d_i/|D|$
Accession number of document D
Normalized weight of term i in D

Given a query vector $Q = (q_1, q_2, \ldots, q_v)$ where q_i is the weight of term i, document correlations are accumulated one term at a time in the following way:

$$\mathrm{COS}(Q,D) = \sum \left(\frac{q_i}{|Q|}\right)\left(\frac{d_i}{|D|}\right) \qquad \text{(IX-1)}$$

from query vector

from inverted directory entry

Since each directory list contains data pertaining to many documents, sufficient core storage must be available to hold the partial sums related to each document. After accessing all appropriate lists and accumulating sums as illustrated, correlations are sorted and citations retrieved and printed in decreasing order of similarity. A complete example of this process is given in Figure II-3.

The precision-recall data for a search using an inverted file is the same as that for a full search since correlations are computed for all documents D such that $\mathrm{COS}(Q,D) > 0$. The number of correlations and disk accesses is much less than for a full search. In particular, the number of accesses is proportional to the query length (directory scan) and the number of documents the user wishes to view. The remainder of this section considers only the I/O activity in the directory scan, however.

In order to determine how directory scan time varies with query length, a simulation is made similar to those in Chapter VII. That is, disk storage map is constructed containing the track and cylinder locations of inverted directory records as though they were actually stored on disk. During the simulated searches, the number of disk accesses is calculated from the changes in track and cylinder locations as directory lists are obtained. Averaging this data and plotting it versus query length shows the desired relationship between I/O activity and the number of query terms. The storage map is made assuming either (1) indexed sequential access (ISAM) to records or (2) direct access to records, for example, through a scatter storage scheme. In either case, each track of the simulated disk is treated as one physical record and the inverted lists (records) are packed onto it using the storage algorithm in Section III.6. This algorithm balances the amount of wasted disk space and the splitting of records on an unnecessarily large number of tracks (thereby requiring extra accesses during retrieval). Where necessary, space is traded for time only if the waste is less than a threshold amount (θ bytes per track). The parameters for the storage algorithm are shown in Table IX-1 and are the same as those used in tests with clustered files.

The size of each inverted directory list is calculated from the number of uses of the corresponding term. It is assumed that 4 bytes of storage are sufficient to hold a document accession number and normalized weight and that of each list is preceded by 20 bytes of header information. Thus, since term number 4155 appears in 138 documents, its directory list requres 20+4*138 = 572 bytes. When the storage map is actually made, directory records are stored in alphabetical order by keyword. In the case of ISAM access, cylinder and track indices are interspersed at appropriate locations. Table IX-2 shows storage statistics for the

Number of data tracks/cylinder	18
Track size	7294 bytes
Threshold for wasted space (θ bytes/track)	720
ISAM index size (track or cylinder)	300 bytes
Direct access index size	0

Parameters for the Management of the Inverted Document File

Table IX-1

Property	1400 Documents	14000 Documents
Number of inverted directory lists (records)	5030	5030
Storage for record header (bytes)	20	20
Storage for accession number and weight (bytes)	4	4
Average total record size (bytes)	80	620
Total directory size (tracks)	57	432
Percent of disk space wasted	1.8%	1.3%
Percent of records requiring an extra access (100% = 5030)	0.04%	3.3%

Results of Storing the Inverted Directory

Table IX-2

inverted directories of two collections. The smaller collection represents the actual Cranfield documents. The larger collection approximates what the Cranfield collection might look like after a tenfold size increase. Assuming each term continues to occur with its present relative frequency, each list is simply 10 times larger than before. This is somewhat tenuous when applied to individual terms, but might be an adequate overall approximation. The new terms that would enter the vocabulary are not considered here. The table data shows only a 1%-2% waste of disk space for either collection even though up to 10% is allowed (720 bytes/track). Because the inverted lists are longer in the larger file, the storage algorithm splits a greater percentage of them over an unnecessarily large number of tracks. This increases retrieval time significantly since the lists which are split usually correspond to frequent document and request terms. Overall, the inverted directory for the 1400 Cranfield documents requires 57 disk tracks. Using the complete combined file takes another 62 tracks for document vectors (citation data, terms, and weights). If relevance feedback and space modification are not desired, the consecutive file could be limited to citation data only. For the Cranfield collection this requires about 14 disk tracks so the total storage space under the inverted organization is either 119 or 71 disk tracks (a 92% or a 15% overhead). By comparison, the entire clustered file in Chapter VII uses 69 tracks (11% overhead) and provides for feedback, space modification, and more flexible searches.

To collect data on the I/O activity during the scan of the inverted directory, simulated searches are made using the Cranfield query set. For each query, the disk arm is "positioned" just outside the directory and track and cylinder changes are monitored as each directory list is accessed. For the ISAM case, the first access picks up the cylinder and first track indices. Thereafter, moving to a new cylinder includes an additional access to obtain the necessary track index. For the case of direct access, it is assumed that either no indices are required or that they are permanently core resident. This may or may not reflect a situation that can actually be implemented. The intent is simply to discover how much overhead is involved in obtaining the ISAM indexes. In both cases one optimization technique is employed. Because the query vector identifies all the directory lists to be accessed, it is assumed that the lists are obtained in a single sweep across the disk surface rather than by jumping forward and backward. During the simulated searches, the evaluation data collected is:

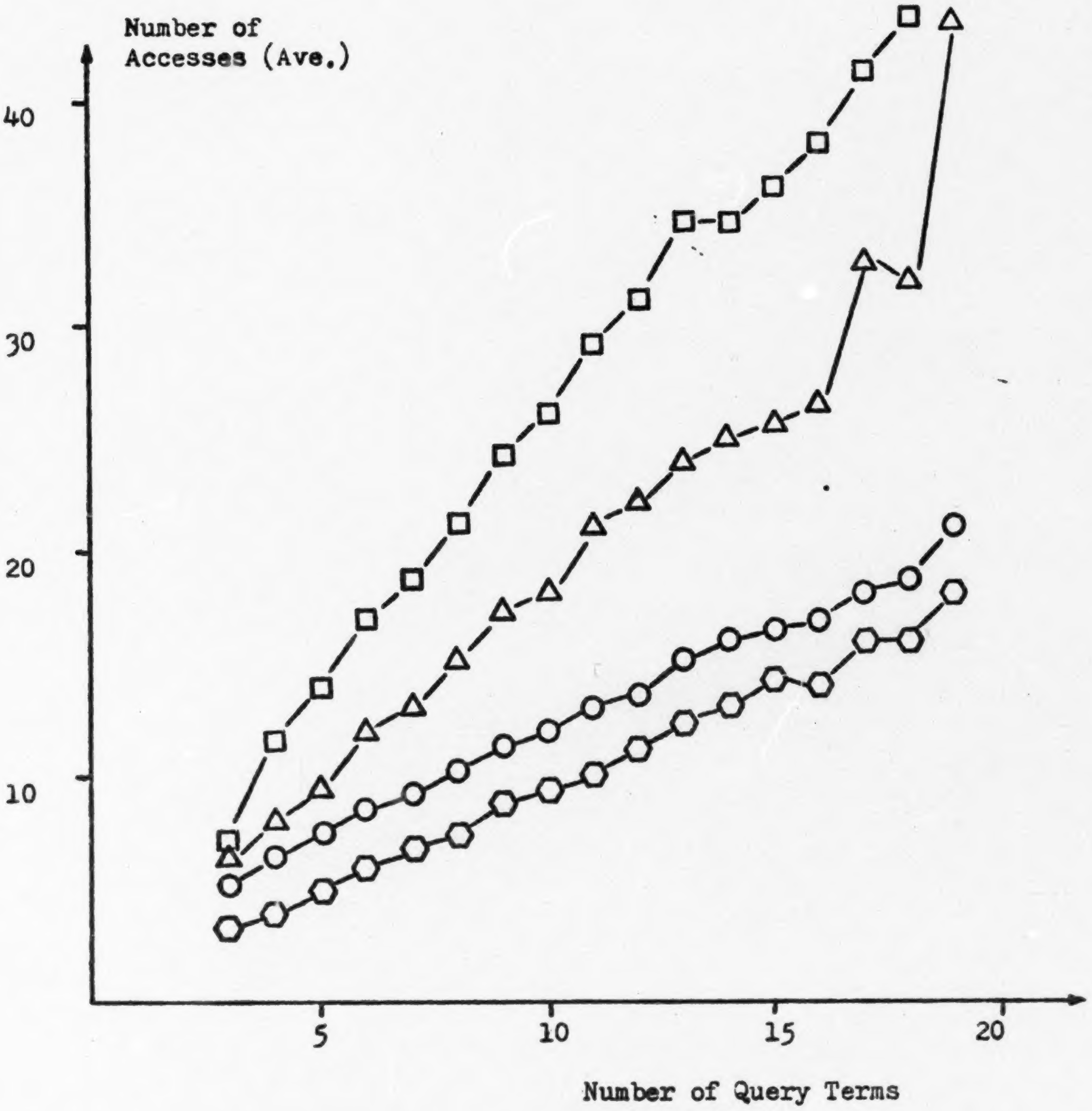
a) the average number of accession for queries of a given length;

b) the average number of track and cylinder changes for queries of each length; and

c) the average stepsize per change (number of tracks or cylinders traversed per change).

Figure IX-1 shows how the average number of disk accesses in an inverted directory scan varies with query length for both the large and small collections. The tests use all 225 requests; however, only a few requests have more than 13 terms, so the points at the right hand end of the curves are somewhat less reliable than the others. In all cases there is a linear relationship between query length and the average number of directory accesses. For 1400 documents, each query term requires 0.9-1.0 disk accesses (average); overall, a penalty of 2.7 accesses is paid for using ISAM. The amount of I/O increases considerably for the larger collection, roughly doubling when 10 times as many documents are added. In this case each term results in 1.6-2.4 accesses and the ISAM overhead is 4-10 accesses. Other collections probably exhibit a similar linear relationship as well as I/O requirements which increase as those observed here. It must be remembered, however, that these figures apply only to the directory scan and not the entire retrieval process. In actual searches, additional accesses are required to obtain and print citation data.

The fact that scan time in an inverted directory increases with query length has good and bad aspects. On one hand, it supplies a convenient and reasonably equitable scheme for recovering search costs, namely by charging a fixed dollar amount for each query term. On the other hand it is difficult to obtain an inexpensive high-precision search, since more accurate searches generally require a moderate number of query terms and thereby incur greater costs. This is especially unfortunate when relevance feedback is used since this process expands a request considerably. Consequently feedback searches could become quite costly. From the users standpoint, it is more satisfactory to separate query formulation and search conditions. For example, a users first task should be to prepare a complete, accurate statement of his information needs. Only then should he consider the amount



Symbol	Access Method	Number of Documents	Average Accesses
□	ISAM	14,000	23.3
△	Direct	14,000	16.6
○	ISAM	1,400 (Cranfield)	10.9
⬡	Direct	1,400 (Cranfield)	8.2

Inverted Director I/O (Disk Accesses) as a

Function of Query Length

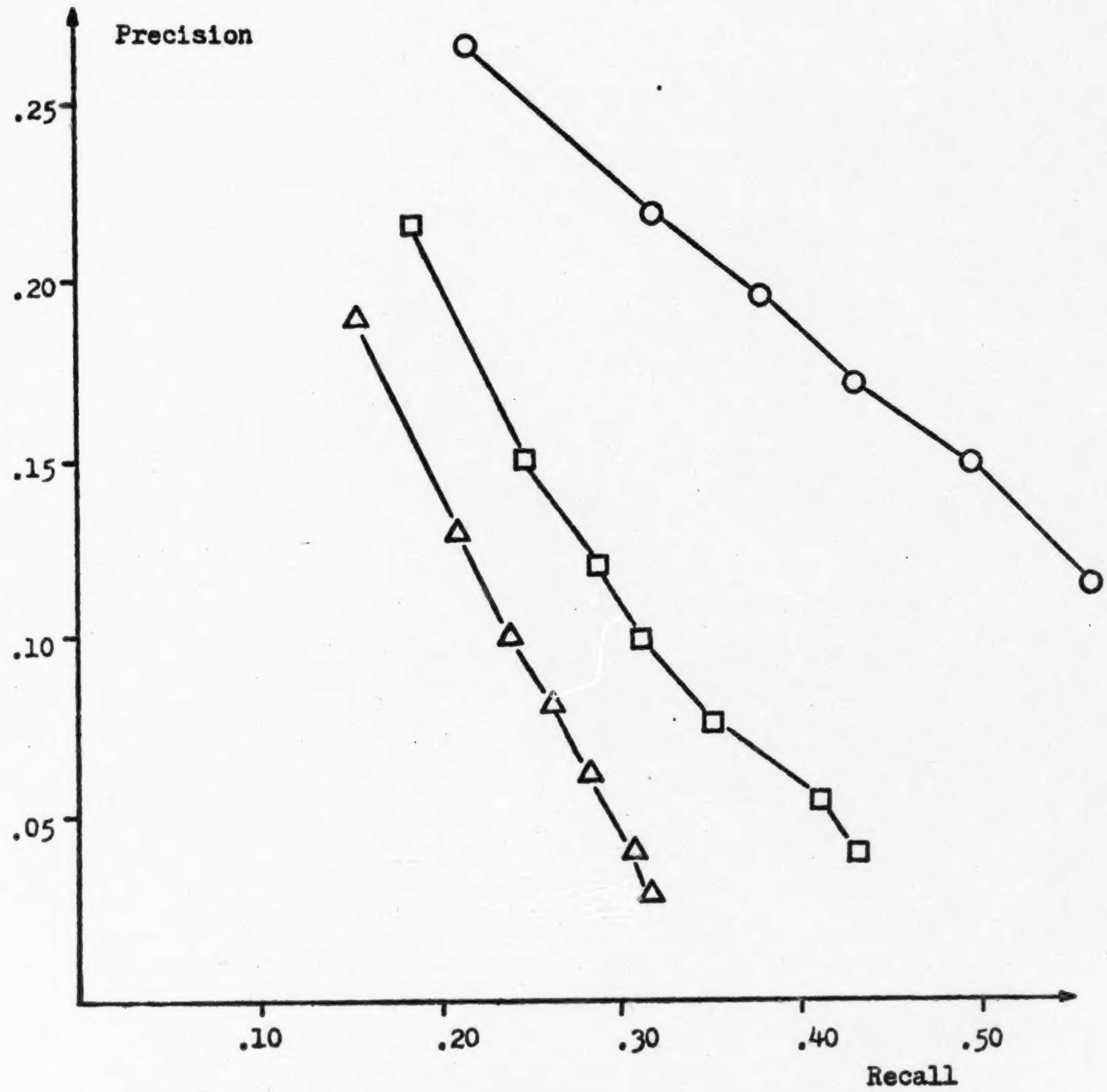
Figure IX-1

of desired output, dollar cost, and other constraints on searching. Unfortunately, this type of separation is difficult to provide with an inverted file since the query formulation controls the search strategy, to a large degree.

3. Comparison of Inverted and Clustered Document Files

This chapter section compares inverted and clustered document files with respect to search speed and quality of retrieval. The previous section considers storage requirements and shows that the inverted scheme needs about twice the space as the clustered scheme if a combined file is used. Regarding quality of retrieval, the precision-recall values from an inverted file search are the same as those for a full search. In a clustered organization, precision-recall data vary with the search strategy; comparisons here are based on the narrow and broad searches of Hierarchy 1 ($P^*_3(\delta = -1)$ profiles) as described in Chapters IV and V. Figure IX-2 shows P-R plots for these searches; following previous practice, the points depict document level averages computed at cutoffs of 5,10,15,20,30,50, and 75. It is seen that at every point, the inverted curve lies significantly above the curve for either cluster search.

What remains is to determine the I/O delay associated with each point of the curve and to evaluate the combined sets of data. To obtain data on I/O activity, searches are simulated for the clustered and inverted files as described in Chapter VII and Section IX.2, respectively. It must be emphasized that even though storage and retrieval are simulated, the experimental parameters are based on properties of the actual Cranfield collections, SMART search system, and IBM 2314 Disk Storage Facility. It is believed



Symbol	Search Description
○	Inverted File (Same as Full Search)
□	Clustered File--Broad Search
△	Clustered File--Narrow Search

Comparison of Precision-Recall Data From
Inverted and Clustered File Searches

Figure IX-2

the results accurately predict actual searches made under the specified conditions.

Since a comparison among file organizations is difficult to make, it is helpful to state the underlying assumptions.

General Assumptions

a) ISAM access to all data items. Both organizations would benefit from direct access, i.e. use of absolute disk addresses, but it is unrealistic to assume use of direct access in operational systems.

b) freshly constructed files. This implies all directory lists and clustered items are physically contiguous in storage rather than having "tacked on" elements due to updating.

The following discussion describes additional considerations for the individual organizations. The I/O data collected in the simulations are summarized in Table IX-3 along with the P-R data shown previously. It will be helpful to refer to this summary as the discussion proceeds.

First, consider the inverted search. The number of disk accesses in the directory scan depends on the query length; a typical query contains 9 terms and therefore requires 11.3 accesses (See Figure IX-1). The I/O for retrieving citations depends on the organization of consecutive file and on the number of citations printed. Recall that the consecutive file in the inverted organization consists of complete document vectors, whereas only the citations are required at the end of a search. Consequently the file might actually consist of two distinct sub-files--one for citations and one for terms and weights. The higher density of citations per track results in fewer disk accesses when the subfile scheme is used. Table IX-3 includes data for both approaches. Another problem in collecting this data is caused by the fact that the identifiers of all retrieved, non-relevant documents are unknown. To circumvent this situation all documents are assumed to have an equal probability of being retrieved and data is based on random selections from the file. Even under these conditions, it is quite probably that the simulated I/O is close to its true value. As an example from the table, consider a typical Cranfield request having 9 terms and 7 relevant documents. A search retrieving 10 documents (2 relevant 8 non-relevant) would require a total of 19 or 24 accesses depending on whether a citation subfile is used. In either case, 11 accesses are spent in the directory scan and the remainder in obtaining document citations (here, arbitrarity selected). A summary of the additional assumptions for the inverted file search includes

a) 9 terms per request (the average for the Cranfield queries);

b) sufficient core storage to hold all partial correlations during the directory scan;

c) equi-probable retrieval of all documents.

The cluster search assumes the same file and search parameters as in Chapter VII, namely Hierarchy 1, $P_3^*(\delta = -1)$ profiles stored by levels, and the narrow and broad strategies used throughout the research. As shown in the table, the narrow search requires an average of 9.7 accesses per query for correlations while the broad search uses 15.4 accesses per query. In many cases, enough core storage is available to contain the citations of

Inverted File Search

Retrieval Cutoff	Recall	Precision	Directory Accesses	Total Accesses/Query 2-Subfiles	No Subfiles
5	.216	.266	11.3	16	19
10	.318	.218	11.3	19	24
15	.377	.186	11.3	21	28
20	.429	.172	11.3	22	31
30	.493	.148	11.3	24	38
50	.563	.126	11.3	25	48
75	.634	.144	11.3	26	57

Clustered File - Narrow Search

Retrieval Cutoff	Recall	Precision	Correlation Accesses	Total Accesses/Query No-rescan	Rescan
5	.156	.190	9.7	9.7	16
10	.208	.130	9.7	9.7	16
15	.239	.102	9.7	9.7	16
20	.261	.083	9.7	9.7	16
30	.284	.063	9.7	9.7	16
50	.307	.042	9.7	9.7	16
75	.316	.029	9.7	9.7	16

Clustered File-Broad Search

Retrieval Cutoff	Recall	Precision	Correlation Accesses	Total Accesses/Query No-rescan	Rescan
5	.185	.216	15.4	15.4	27
10	.245	.151	15.4	15.4	27
15	.286	.121	15.4	15.4	27
20	.311	.101	15.4	15.4	27
30	.349	.077	15.4	15.4	27
50	.406	.055	15.4	15.4	27
75	.429	.039	15.4	15.4	27

Comparison of I/O Requirements in Inverted and Clustered File Searches

Table IX-3

all documents to be retrieved, thereby avoiding a rescan of level-3 items. For example, if the user requests 30 documents, the search program might always maintain in core the citations of the 30 documents with the highest correlations. As the search progresses, higher scoring documents are added to this "active" list and others are deleted. If memory space is limited, just the highest correlations need be kept along with the corresponding document accession numbers. At the end of the correlation phase, citations are obtained from disk. In most cases, this rescan requires a considerable number of additional disk accesses (6.3 or 11.5) accesses depending on the search strategy). For example, suppose 5 clusters are expanded, resulting in 150 document correlations. If 30 documents are returned to the user, they will undoubtedly come from all 5 clusters. Hence fetching their citations involves about the same amount of I/O as the initial scan of level-3. Consequently an additional economy in a clustered file is realized, when a reasonable amount of core storage is available. Data for both cases--no rescan and rescan--is shown in Table IX-3. It would appear that a similar procedure would apply to the inverted search. This is not the case, however, since correlations are accumulated and none can be discarded until the very end. Saving all citations would require a prohibitive amount of core storage.

Contrasting the I/O requirements reveals that the inverted search obtains its superior P-R performance with significantly more disk accesses than a cluster search. Consequently, the response time to an on-line user is expected to be greater. However, the clustered file provides either high recall or low recall searches for approximately the same number of accesses. The inverted file gives a single type of search, but at higher precision. Of note is the fact that the scan of the inverted directory or profile hierarchy takes approximately the same effort, depending on the search strategy. The essential difference in the schemes is that at some point, the inverted search makes random accesses into the data base for individual items, for example, to obtain citations. The cluster search also makes random accesses, but only for groups of documents. Its economy results from having concentrated, in a few locations, all documents having a high probability of being relevant. Furthermore this economy is likely to remain or even increase in larger collections. For example, a cluster hierarchy need not grow in direct proportion to collection size, but an inverted directory must increase proportionately in order to maintain updated term entry lists. In addition, a larger collection implies relevant documents are spread over more disk space in an inverted file since documents have more or less arbitrary locations. Clusters, however, retain their high density of relevant to a large extent. These are at least two reasons why a clustered file should be superior on a larger collection also.

However, it cannot be denied that an inverted file gives higher precision searches whereas a clustered file is more economical of storage space and provides more flexible searches. The ideal situation is to combine both schemes, i.e. provide an inverted directory to clusters of documents. This might be feasible if the directory size could be reduced to 10% of collection size, for example. The problem, naturally, is to make an accurate differentiation among clusters on so little information. This is exactly the problem considered in Chapter V where profiles are found to provide an adequate solution.

4. Summary

This chapter compares the inverted and clustered file organizations for the Cranfield collection on the basis of storage requirements, search speed, and quality of retrieved output and attempts to generalize the findings to larger document collections. As depicted here, the inverted organization consists of a directory and consecutive files with a storage overhead of 15% or 92%. The latter figure applies if relevance feedback or space modifications is included in the system. The clustered file incurs an 11% storage overhead.

The search time (cost) and quality of retrieved output are harder to compare because they are interdependent. For inverted files, search time is a function of query length and the number of retrieved documents. For a given query, the directory scan takes a fixed number of disk accesses (about 1 access per term for the Cranfield collection). Thereafter, the search cost and precision-recall values are determined by the number of documents retrieved. (See Tables IX-2 and IX-3.) Presumably high precision or feedback searches are expensive since they involve requests having many terms and thus spend a lot of time in the directory scan. High recall searches are costly because they generally require accessing a large number of documents from arbitrary disk locations. The cluster search uses a number of disk accesses proportional to the number of expanded clusters. Search time is not dependent on the length or complexity of a request, but on the broadness or narrowness of the search strategy. Profile correlations are an overhead cost and serve to select several general areas of the disk from which to begin retrieving documents. Once this phase is completed, a search at any recall level may be obtained, for about the same cost, simply by retrieving additional documents. The overall precision is less than in the inverted search, however.

As noted earlier, the economics of the clustered file organization should be present and perhaps more apparent in larger files. There is every reason to believe that the profile hierarchy grows slower than the inverted directory, since each new document does not necessarily increase the size of any profile. In addition, with limited re-clustering, it should be possible to maintain reasonable groupings of relevant documents and, therefore, quick, successful searches.

Chapter X

Summary, Conclusions, Discussion, and Suggestions for Future Work

There are several ways to summarize this research and its importance to information retrieval. A simple listing of chapter contents is a good starting point.

Chapter	Contents
1	Introduction to document retrieval systems, automated text processing, search techniques, file organization, and disk storage devices
2	Survey of logical file organizations (sequential, chained, inverted, calculated access, clustered) and physical organizations (serial, direct, indexed sequential)
3	Detailed description of clustered files including classification schemes, hierarchy structure, profile definition, search strategies, updating procedures, storage considerations, query clustering, and alternate uses of the cluster hierarchy (underlined topics are the basis of experiments in later chapters)
4	Development of evaluation procedures; description of the experimental collections
5	Experiments with profile definition, specifically examining standard and rank value profiles, search bias, vector length, frequency considerations, as

Chapter	Contents Cont'd.
	well as unweighted and partially weighted vectors
6	Experiments related to updating a clustered document collection (profile maintenance schemes and rate of hierarchy degeneration)
7	Experiments with schemes for storing the hierarchy on disk in conjunction with ISAM indices; development of a disk storage and retrieval simulation model
8	Experiments with automatic query alteration using information within the profile hierarchy
9	Comparison of clustered and inverted files with respect to storage, speed, and quality of retrieved output
10	Summary and conclusions

The results of this work are applicable to various types of research in information retrieval. Within the SMART project, these efforts produced a new document collection--the Cranfield 1400 stem. The parameters for the collection preparation and a summary of its properties are contained in Chapter IV and Appendix A. In addition, Chapter VI contains an algorithm for splitting a collection into special test subcollections needed for these and other experiments (see Appendix B).

On a higher level, some of the genuine contributions of this work reside in its test and evaluation methods. The cluster-oriented evaluation scheme developed in Chapter IV and used throughout is unique in that it is independent of search strategy, accounts more accurately for system effort (disk accesses), and cost relatively little to run. Chapter V introduces the concept of biased search results and sets forth a procedure for detecting bias with respect to a given profile property. The technique can be extended to other properties and many types of searches (full, inverted, etc.) and should be useful in many experimental setups. Chapter VII utilizes an interesting disk storage algorithm and simulation scheme for examining questions related to search speed and space utilization. Finally, the work in Chapter VIII includes a new way of computing similarity matrices, namely by summing matrices for subsets of desired items.

A large number of results are directly related to the use of clustered document files. These are explained in detail and adequately summarized in Chapters V to VIII. Only a general summary will be given here. First, the experiments in Chapter V show that it is possible to make a reasonably accurate, economical cluster profile, which is free from correlation domination and bias. Specifically, its term weights should be based on frequency ranks and therefore be non-decreasing with the number of occurrences. A large number of low weighted terms can be deleted with a considerable saving in storage space. Second, the most satisfactory update procedure in a clustered file alters the weights of only existing profile terms. Even so, the hierarchy degenerates with the addition of new items and should undergo at least partial re-clustering when it increases 25%-50% in size. (This percentage is figured as the ratio of additions to the current file size.) Third, a cluster hierarchy should be stored by levels to facilitate rapid searching. Finally, term substitutes within profiles should not be used to automatically alter requests as described in Chapter VIII. However, there are other alternate ways of using the cluster hierarchy which can help justify the expense of document classification.

These results do not suggest that no further improvements might be made in profile definition. For example, a large discrepancy remains between the best achievable performance curve (not the ideal curve) and those actually obtained in these experiments. It is felt that additional improvements can be made, perhaps by using partial weighting techniques or a completely different scheme altogether. Some problems undoubtedly are related to the document indexing. If changes are made in profiles, the optimal updating scheme may change also and the general hierarchy quality might become more sensitive to new additions. With less speculation it can be said that there is a genuine need for developing new, additional uses for a cluster hierarchy. A few are suggested in Chapter III; for example, using a hierarchy in selective dissemination of information and document browsing.

In the final analysis, this investigation attempts to answer the question "Is a clustered file organization suitable for on-line document retrieval?". Part of the answer is obtained from the comparison with inverted organization in Chapter IX. A clustered file is found to compare favorably in terms of search speed and storage economy. Search precision is less, but compensated by a flexible level of recall (low or high). In general, the clustered organization provides a great deal of flexibility, allowing any type of request-document matching, search strategy, or feedback. Part of this is due to the fact that the entire document remains intact in storage, rather than being split up and stored in pieces. Thus all information is available for use by matching coefficients, feedback schemes, etc. Furthermore, it is certain that on-line retrieval must move away from making arbitrary accesses into a data base for individual records. A clustered file solves this problem by concentrating, those records with a high probability of satisfying a request, in a few disk areas. Therein lies its greatest value.

Appendix A

Common Word List

The forms of the Cranfield document and query collection used in this research are produced by removing common words from the original document and query texts and by applying a analysis scheme to reduce variants of a word to the same stem. The common word list (restriction list) includes 360 prepositions, pronouns, conjunctions, and verbs of the following types.

1. Prepositions (of, on, at, in, . . .)
2. Pronouns
   a) Personal (he, she, they, . . .)
   b) Possessive (his, hers, my, . . .)
   c) Reflexive (myself, herself, . . .)
   d) Interrogative (who, which, . . .)
   e) Demonstrative (this, that, . . .)
   f) Indefinite (all, any, both, each, many, . . .)
3. Conjunctions
   a) Coordination (and, but, or, . . .)
   b) Correlative (either, whether, not only, . . .)
   c) Subordinating (after, because, how, unless, . . .)
4. Verbs
   a) Auxiliaries and their forms (be, do, have, can, may, . . .)
   b) Non-content (begin, choose, make, come, give, keep, meet, put, say, see, show, take, . . .)

5. Other
    a) Individual letters
    b) Punctuation
    c) Numbers

The suffix removal program uses the standard SMART suffix list and will not be described here.

Appendix B

Subcollections for the Updating Experiments

In order to conduct the updating experiments in Chapter VI, the Cranfield documents are separated into subsets with special properties. Initially the 1400 documents are divided into halves--sets A and B--each containing documents chosen in such a way that half the relevant for each query lie in each subset. Then, the B subcollection is split halves again--set C and D--each containing one quarter of the total relevant for each query. The documents in sets A, C, and D are listed below; the algorithm by which they are derived is given in Chapter VI.

A subcollection

2 4 6 8 10 13 15 17 19 21
24 26 28 30 32 34 37 39 41 43
44 46 48 50 52 54 56 59 62 63
64 65 67 69 70 72 74 76 78 80
82 84 86 88 90 92 94 96 97 99
101 103 105 107 109 111 113 114 116 120
122 124 126 129 131 133 135 137 140 142
143 146 148 151 153 155 156 157 159 161
163 165 167 169 171 173 175 177 179 181
183 185 187 189 192 194 196 199 201 204
206 208 210 212 214 216 218 220 223 226
228 231 233 236 238 240 241 243 245 247
250 252 254 257 259 262 264 267 269 271
273 275 277 279 281 283 285 287 289 291
293 295 297 299 300 302 304 306 308 310
312 315 317 319 321 323 325 327 329 331
334 336 338 340 342 345 347 349 351 353
355 357 359 361 363 365 367 369 371 372
374 376 378 380 382 384 386 387 389 390
393 395 397 399 401 403 406 408 410 411
411 415 416 418 420 422 424 426 428 430
432 434 437 439 441 443 445 447 449 452
454 456 458 460 462 464 466 468 470 472
474 475 477 478 479 482 484 486 487 489
491 492 494 496 497 500 503 504 506 508
510 512 514 516 518 520 522 524 526 528

A Cont'd.

530 532 534 536 538 540 541 543 545 547
549 550 552 555 557 559 561 563 565 567
571 573 574 575 577 579 581 583 585 587
589 592 593 596 597 599 601 603 605 607
609 613 615 617 619 621 623 625 627 630
632 634 636 639 641 643 645 647 649 651
654 655 657 659 660 661 663 665 667 670
672 674 677 679 681 683 686 690 692 694
696 698 700 702 705 707 708 710 711 713
716 718 721 723 725 727 729 731 733 735
737 739 741 743 746 748 750 752 754 756
758 760 762 763 765 766 768 770 772 774
776 778 780 782 784 786 788 790 792 794
796 798 800 802 804 806 808 810 812 814
816 817 819 821 823 825 827 828 829 831
833 835 837 839 841 843 845 847 849 851
853 854 856 858 860 862 864 866 868 870
872 874 876 877 879 881 883 885 887 889
891 893 895 897 898 900 902 904 906 909
911 913 914 917 919 921 922 924 925 928
930 931 933 935 937 939 941 943 945 947
949 951 953 955 957 959 962 963 964 966
968 970 972 974 975 976 979 980 982 984
986 988 990 993 995 996 997 999 1001 1003
1005 1007 1009 1011 1013 1015 1017 1019 1022 1024
1026 1028 1031 1033 1035 1036 1039 1041 1043 1045
1047 1049 1051 1053 1055 1057 1059 1060 1063 1065
1067 1069 1070 1071 1073 1075 1077 1079 1081 1083
1085 1087 1089 1091 1093 1095 1097 1099 1101 1103
1105 1107 1110 1112 1114 1116 1118 1120 1122 1124
1125 1127 1129 1130 1132 1134 1136 1138 1139 1142
1144 1147 1149 1151 1153 1155 1157 1158 1160 1162
1164 1166 1168 1170 1172 1174 1178 1180 1182 1184
1186 1189 1192 1194 1195 1197 1199 1200 1202 1204
1206 1208 1210 1212 1214 1216 1218 1220 1222 1224
1226 1228 1230 1236 1238 1240 1242 1243 1245 1248
1250 1252 1254 1255 1257 1259 1261 1264 1265 1267
1269 1271 1273 1274 1276 1278 1280 1282 1284 1286
1289 1291 1293 1295 1298 1300 1302 1303 1305 1307
1309 1310 1311 1314 1315 1317 1319 1321 1323 1325
1327 1329 1331 1332 1333 1334 1335 1337 1338 1340
1343 1345 1347 1349 1351 1353 1355 1357 1358 1360
1362 1365 1367 1369 1371 1372 1373 1375 1377 1379
1381 1383 1385 1387 1388 1390 1392 1394 1397 1399

C Subcollection

1 5 9 12 16 20 22 23 25 29
33 35 38 45 49 51 58 61 71 73
77 81 87 91 95 100 104 108 112 118
121 125 130 132 136 141 149 152 160 164
168 174 178 184 188 190 193 197 200 203
207 211 213 216 221 224 225 229 232 235
239 244 245 251 253 258 261 266 270 272
276 280 284 288 290 29[illegible] 298 301 305 307
311 314 318 322 324 328 332 333 337 341
346 350 356 362 366 370 375 379 383 391
394 398 402 404 409 417 419 423 425 429
435 438 444 448 450 453 459 463 467 471
480 483 490 495 501 507 511 515 519 523
527 529 533 537 539 542 546 548 551 554
558 562 566 572 576 580 582 586 590 594
595 598 600 604 608 611 614 616 622 624
628 631 635 638 642 646 650 653 658 664
666 668 671 676 680 685 688 691 695 697
701 704 709 714 717 720 722 726 730 734
738 740 744 747 751 755 759 764 769 773
777 781 785 789 793 797 801 805 809 813
818 822 826 830 834 838 844 848 855 859
861 863 865 869 871 880 884 890 892 899
903 907 910 916 918 923 927 934 938 942
946 950 952 956 958 961 967 971 978 981
985 989 992 998 1002 1004 1008 1012 1016 1021
1023 1027 1030 1032 1038 1042 1046 1050 1052 1058
1062 1066 1069 1074 1078 1080 1086 1088 1092 1096
1098 1100 1104 1106 1111 1115 1119 1121 1128 1133
1137 1141 1146 1152 1156 1161 1165 1169 1176 1177
1181 1185 1190 1196 1201 1205 1209 1213 1217 1221
1225 1229 1232 1234 1241 1246 1249 1253 1260 1263
1268 1272 1277 1283 1288 1292 1296 1299 1300 1306
1313 1318 1320 1326 1330 1339 1342 1348 1350 1354
1359 1364 1366 1374 1378 1382 1386 1389 1395 1398

D Subcollection

3	7	11	14	18	27	31	36	40	42
47	53	55	57	60	66	68	75	79	83
85	89	93	98	102	106	110	115	117	119
123	127	128	134	138	139	144	145	147	150
154	158	162	166	170	172	176	180	182	186
191	195	198	202	205	209	215	219	222	227
230	234	237	242	246	249	255	256	260	263
265	268	274	278	282	286	292	296	303	309
313	316	320	326	330	335	339	343	344	348
352	354	358	360	364	368	373	377	381	385
388	392	396	400	405	407	412	414	421	427.
431	433	436	440	442	446	451	455	457	461
465	469	473	476	481	485	488	493	498	499
502	505	509	513	517	521	525	531	535	544
553	556	560	564	568	569	570	578	584	588
591	602	606	610	612	618	620	626	629	633
637	640	644	648	652	656	662	669	673	675
678	682	684	687	689	693	699	703	706	712
715	719	724	728	732	736	742	745	749	753
757	761	767	771	775	779	783	787	791	795
799	803	807	811	815	820	824	832	836	840
842	846	850	852	857	867	873	875	878	882
886	888	894	896	901	905	908	912	915	920
926	929	932	936	940	944	948	954	960	965
969	973	977	983	987	991	994	1000	1006	1010
1014	1018	1020	1025	1029	1034	1037	1040	1044	1048
1052	1056	1061	1064	1072	1076	1082	1084	1090	1094
1102	1108	1109	1113	1117	1123	1126	1131	1135	1140
1143	1145	1148	1150	1154	1159	1163	1167	1171	1173
1175	1179	1183	1187	1188	1191	1193	1198	1203	1207
1211	1215	1219	1223	1227	1231	1233	1235	1237	1239
1244	1247	1251	1256	1258	1262	1266	1270	1275	1279
1281	1285	1287	1290	1294	1297	1304	1308	1312	1316
1322	1324	1328	1336	1341	1344	1346	1352	1356	1361
1363	1368	1370	1376	1380	1384	1391	1393	1396	1400

Appendix C

Confirmation Test Evaluation Curves

Section 9 of Chapter V summarizes a set of tests confirming that profiles for various cluster hierarchies behave in approximately the same way. That is, the preliminary conclusions drawn from the experiments on Hierarchy 1 are supported by the experiments on Hierarchies 2 and 3. Those preliminary conclusions are:

a) profile term weights based on frequency ranks are superior to those based on frequency counts $P_3^* > P_3$;

b) a large number of low weight profile terms can be deleted without adversely affecting performance $P_3^*(\delta = -1) \approx P_3^*$; and

c) shortened unweighted profiles are roughly equivalent to weighted profiles $P_1^*(\delta = -1) \approx P_3^*(\delta = -1)$.

In accordance with the scheme set up in Chapter IV, the confirmation tests involve both cluster-oriented evaluation (PR-PF data from levels 1 and 2) as well as SMART evaluation P-R data from narrow and broad searches). In each of these four trials, an attempt is made to fairly judge the relative merit of four profile types in the three hierarchies. Thus, there are a total of 12 plots of performance, each showing 4 curves. A consistent notation is used in these plots which is explained in Table C-1 along with some properties of each hierarchy. It is worth remarking that the absolute values of the performance measures differ substantially among the collections. However, the primary concern here is with the relative positions of performance curves and not the actual measured values; hence different scales are of secondary importance.

First, consider the RC-PF plot for any collection and hierarchy level (Figures C-1, C-2, C-5, C-6, C-9, C-10). Corresponding points on the 4 curves represent the same amount of system effort--the average number of disk accesses to expand one additional cluster (amount not shown)--and its placement indicates the resulting recall ceiling and precision floor of the search. Therefore, the curves can be compared on a point-to-point basis and profiles ranked accordingly. Since the curves do not usually overlap, the judgments are made with considerable confidence. Ranks are listed beneath each figure; a composite ranking is shown in Table V-5. As in all comparisons there must be a criteria for determining when curves are significantly different. Here, a 2%-4% difference is considered significant, this amount being about half that used in some SMART experiments. However, 4 times as many queries are used in these tests, so the confidence level for the conclusions remains about the same in both cases.

Accurate judgments are a bit more difficult to make using the SMART precision-recall curves and normalized measures (Figures C-3, C-4, C-7, C-8, C-11, C-12). The basic problem is that of comparing searches involving the same amount of work. Unfortunately, it is impossible to ascertain the number of disk accesses per search. The number of profile and document correlations is available instead and is shown beneath each figure. Small differences in the number of document correlations can be neglected since entire clusters of items are fetched at a time. Differences in the number

Symbol	Description
○	Denotes curves for P_3 profiles--term weights are proportional to frequency counts
⬡	Denotes curves for P^*_3 profiles--term weights are proportional to frequency ranks
□	Denotes curves for P^*_3 P^*_3, but includes deletion of terms with low weights
△	Denotes curves for P^*_1 vectors based on P^*_3
NR	normalized recall
NP	normalized precision
C(x)	average number of correlations on hierarchy level x
RANK	relative evaluation rank of profile types taking into account performance and search effort

a) Notation Used Throughout Confirmation Tests

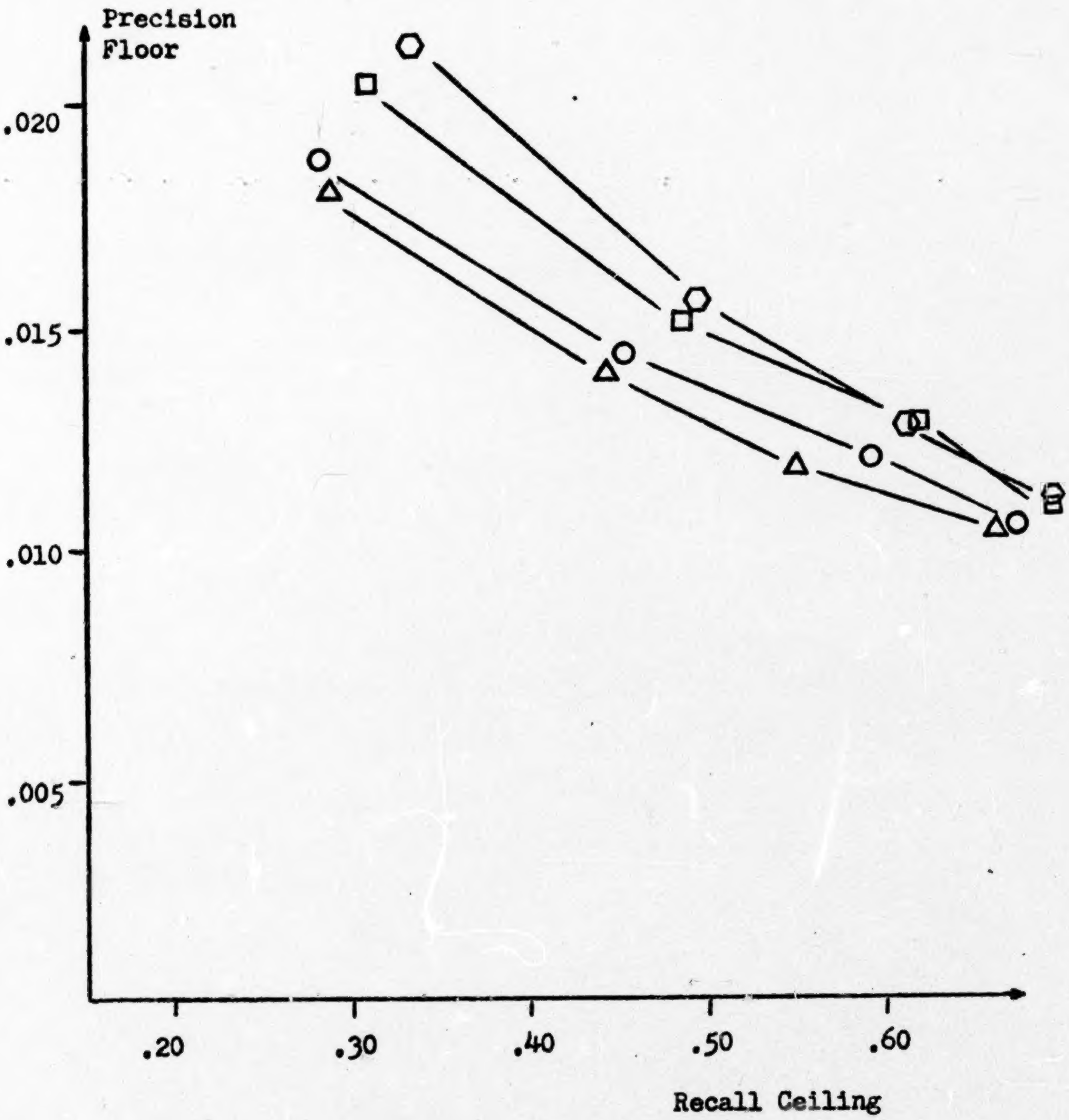
Hierarchy	Level	Number of Profiles	Average Length Before Deletion	Average Length After Deletion ($\delta = -1$)
1	1	13	812	141 (17%)
1	2	55	323	70 (22%)
2	1	6	908	207 (23%)
2	2	94	311	69 (22%)
3	1	28	526	103 (20%)
3	2	103	197	47 (24%)

b) Selected Properties of Profiles

Table C-1

of profile correlations are more serious since these records are obtained from more or less arbitrary disk locations (1 access per node). Consequently, an equal number of profile correlations is more important in determining expenditure of "equal system effort." Obviously, the most desirable profile provides a superior precision-recall curve for the smallest amount of work. As a practical matter in these comparisons, it is necessary to decide not only what performance difference is significant, but also when superior performance (P-R) must be downgraded because of excess search effort. Here a 2% difference in normalized measures or a 4% difference in P-R curves is considered significant and is offset only by one or fewer profile correlations in the next lower ranking search.

The following evaluation curves are presented using these methods for determining the relative merit of each profile type in the indicated collection. A summary of the rankings and a discussion of the test conclusions are given in Section V.9.

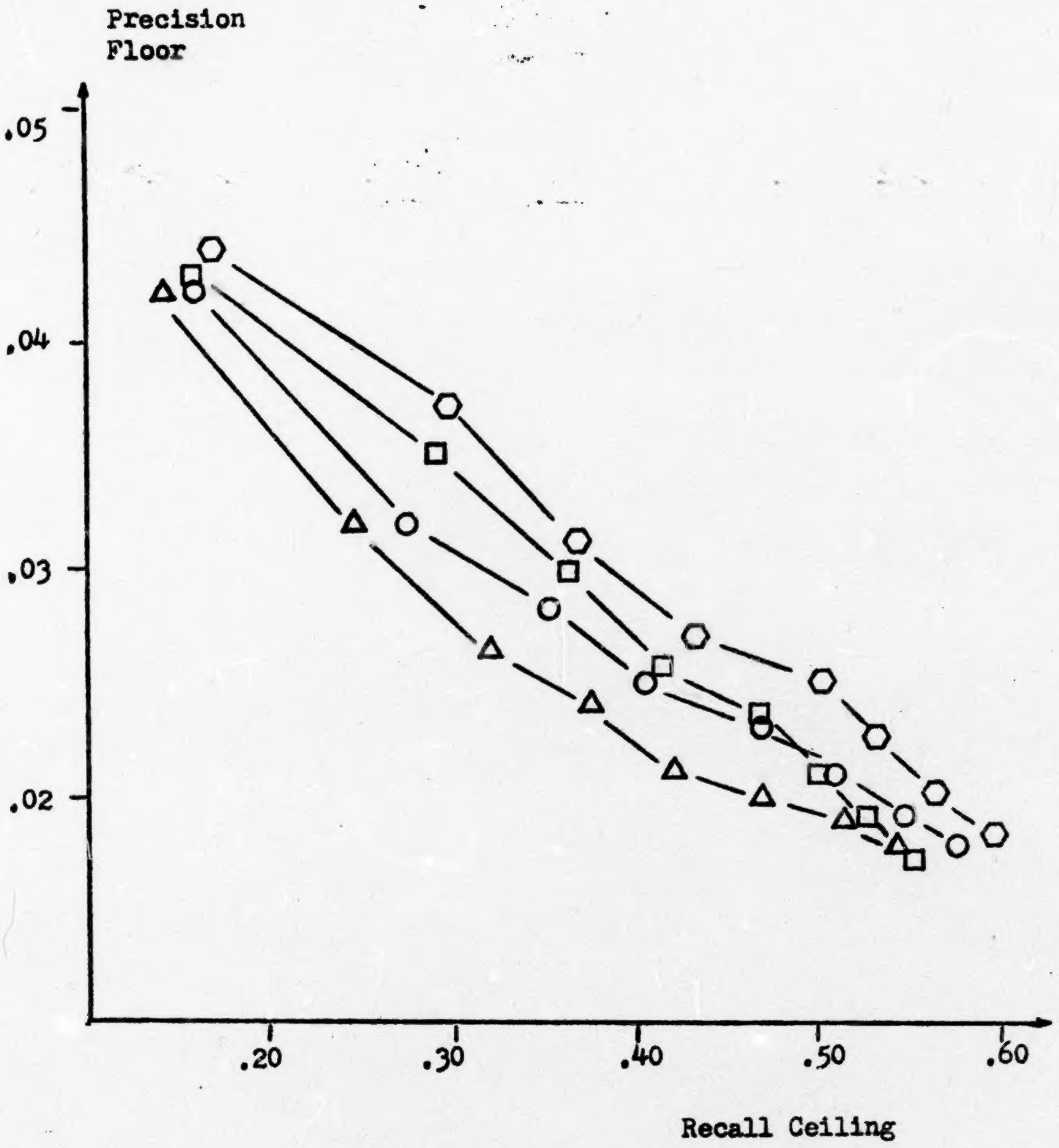


Symbol	Profile Type	Rank
○	P_3	3
⬡	P^*_3	1
□	$P^*_3(\delta = -1)$	2
△	$P^*_1(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation

Hierarchy 1, Level 1

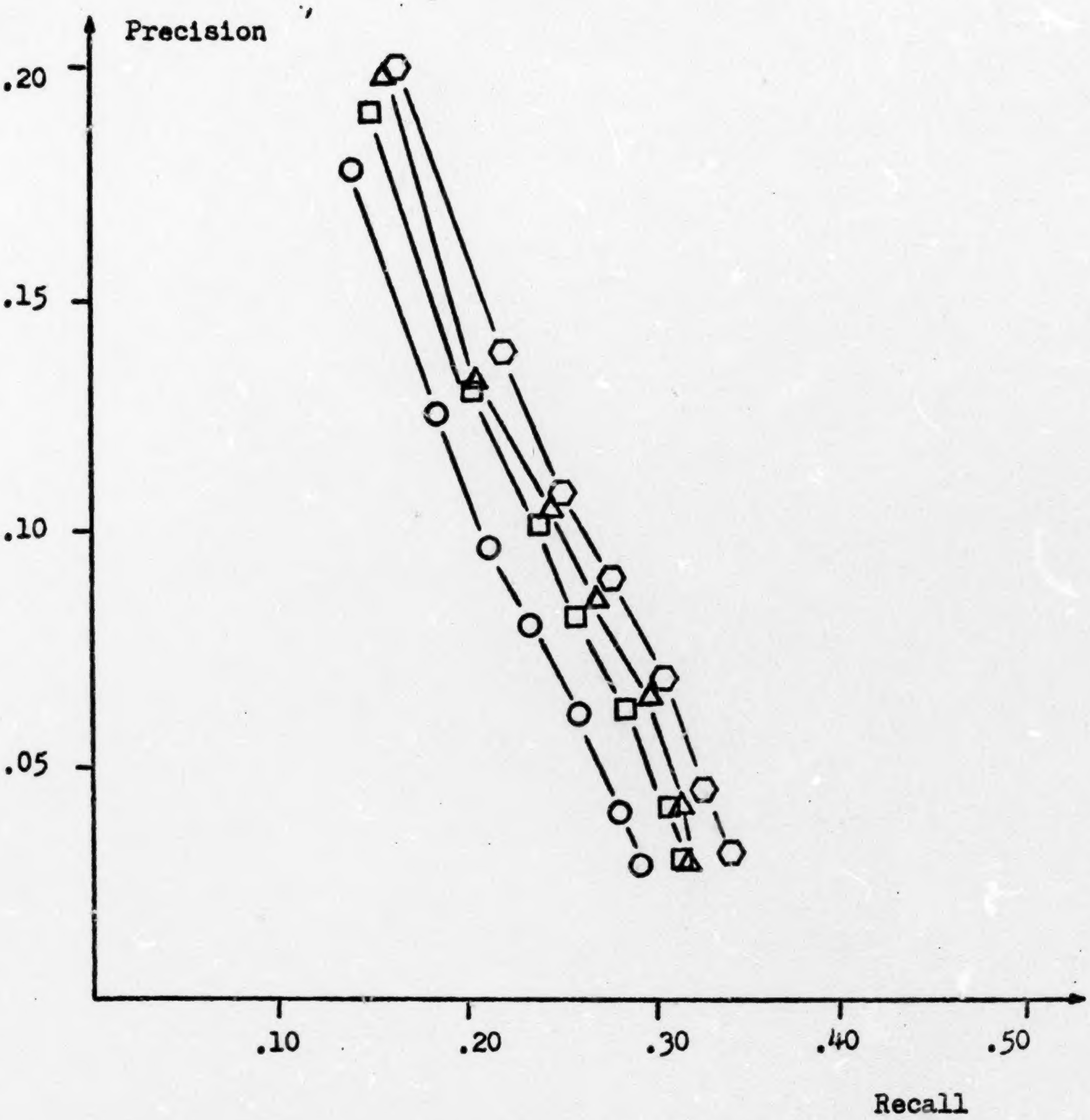
Figure C-1



Symbol	Profile	Rank
○	P_3	3
⬡	P^*_3	1
□	$P^*_3(\delta = -1)$	2
△	$P^*_1(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation
Hierarchy 1, Level 2

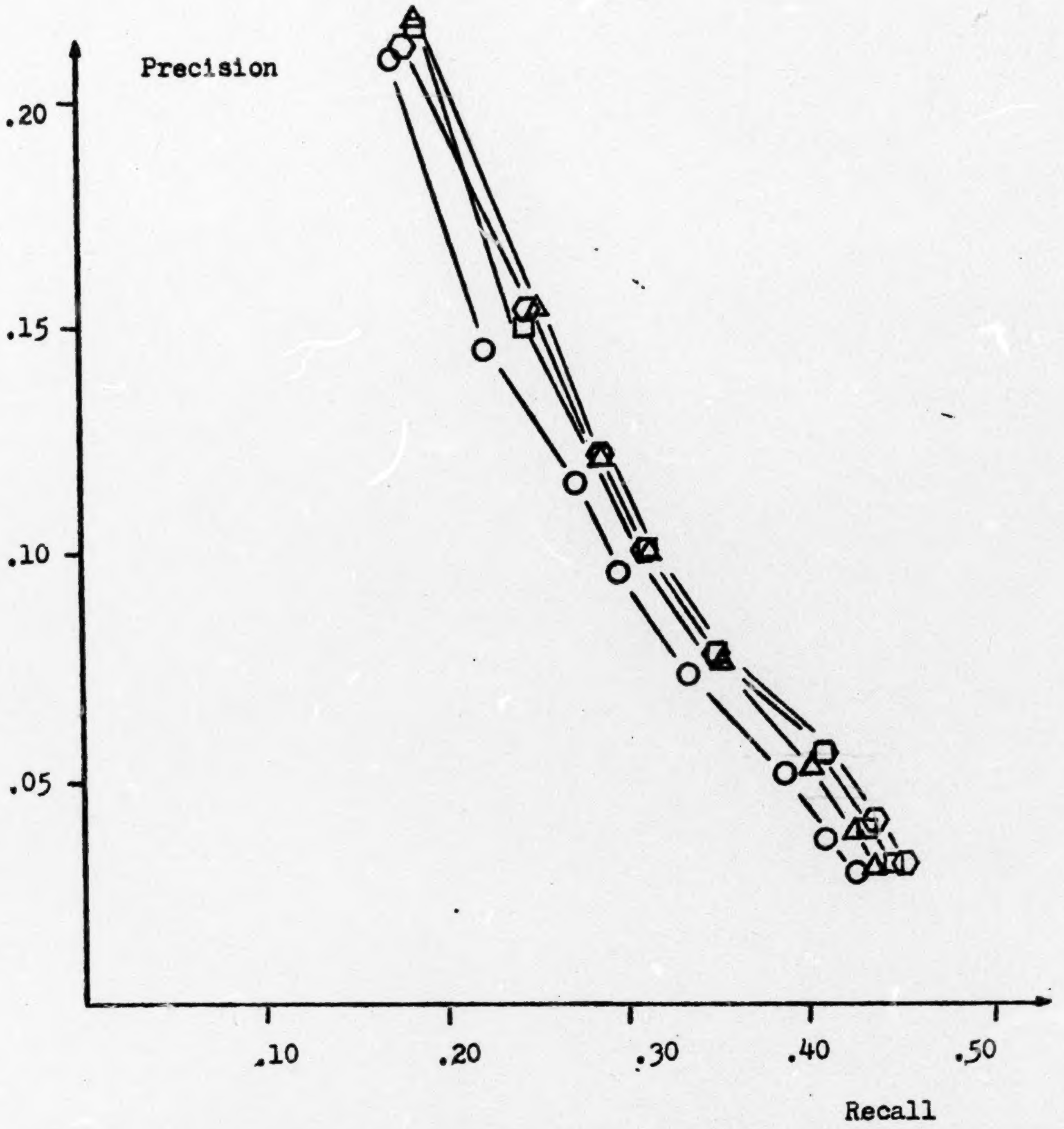
Figure C-2



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.616	.353	13	7.2	90	3
⬡	P^*_3	.644	.390	13	7.5	89	1
□	$P^*_3(\delta = -1)$	.627	.370	13	8.0	86	2
△	$P^*_1(\delta = -1)$	.633	.376	13	9.7	89	4

Confirmation Test Results--SMART Evaluation
Hierarchy 1, Narrow Search

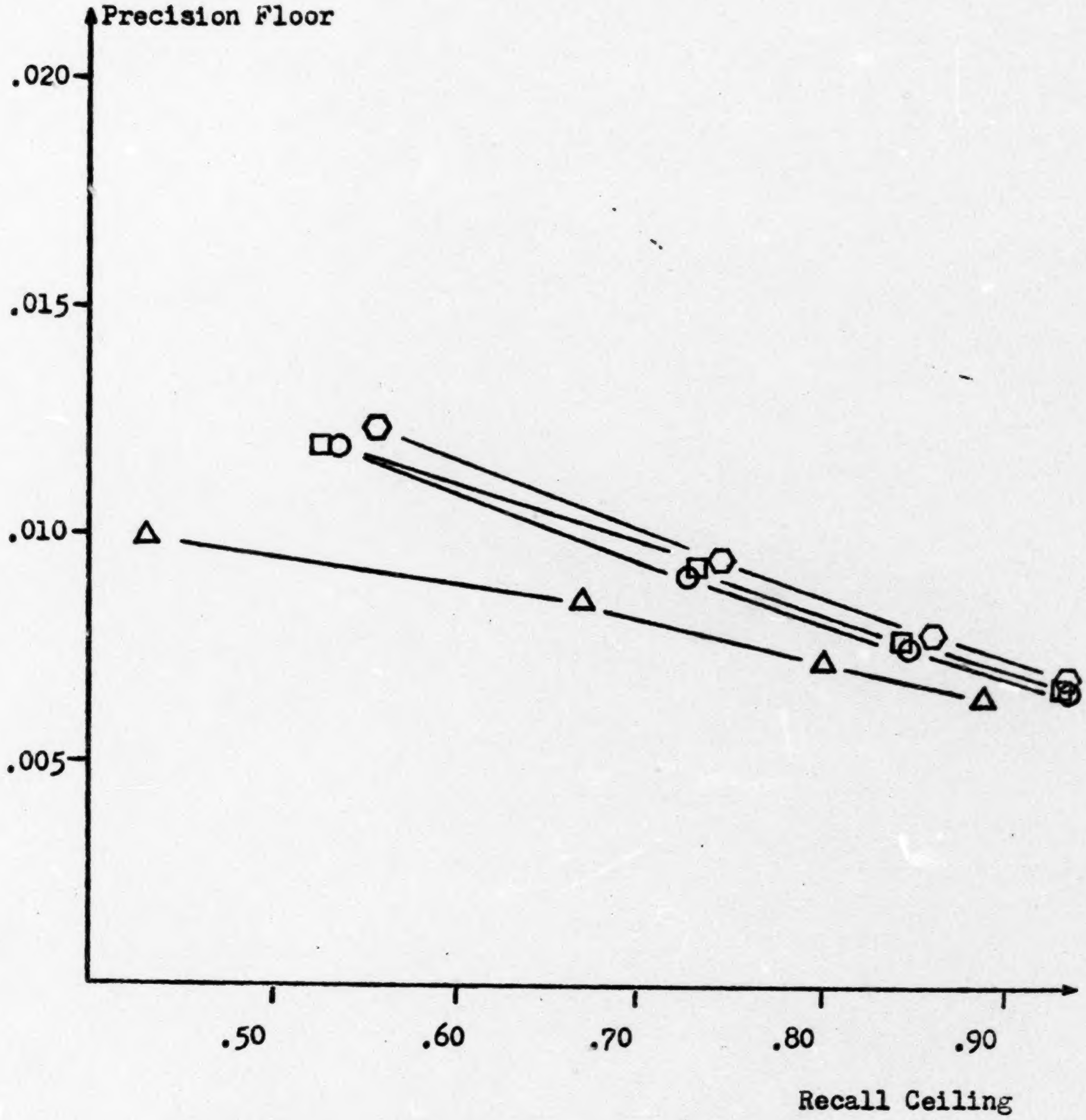
Figure C-3



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.672	.422	13	11.5	160	2
⬡	P^*_3	.689	.440	13	12.1	161	1
□	$P^*_3(\delta = -1)$	.695	.440	13	13.0	162	3
△	$P^*_1(\delta = -1)$	.674	.433	13	14.5	164	4

Confirmation Test Results--SMART Evaluation
Hierarchy 1, Broad Search

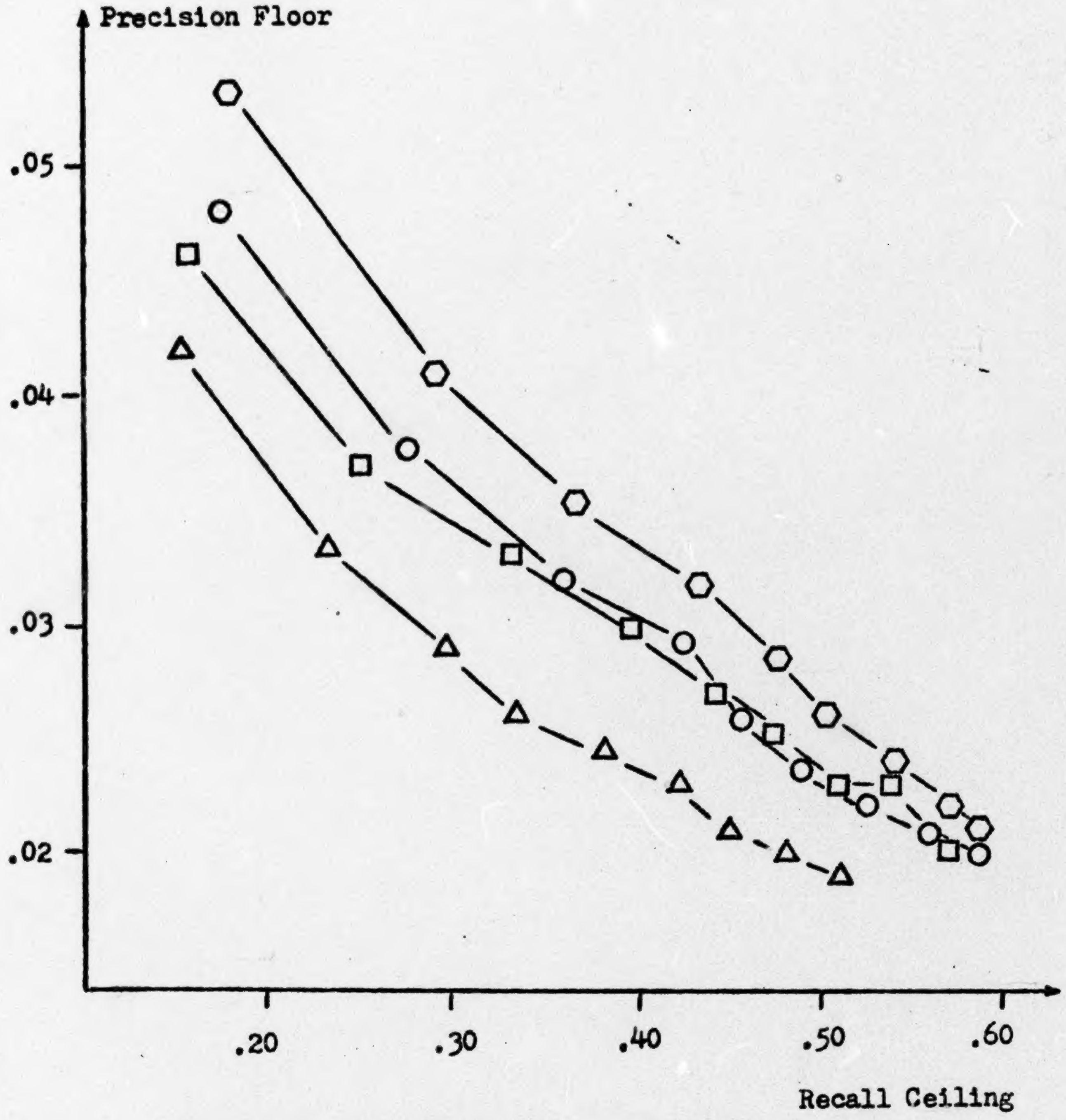
Figure C-4



Symbol	Profile	Rank
○	P_3	2
⬡	P_3^*	2
□	$P_3^*(\delta = -1)$	2
△	$P_1^*(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation
Hierarchy 2, Level 1

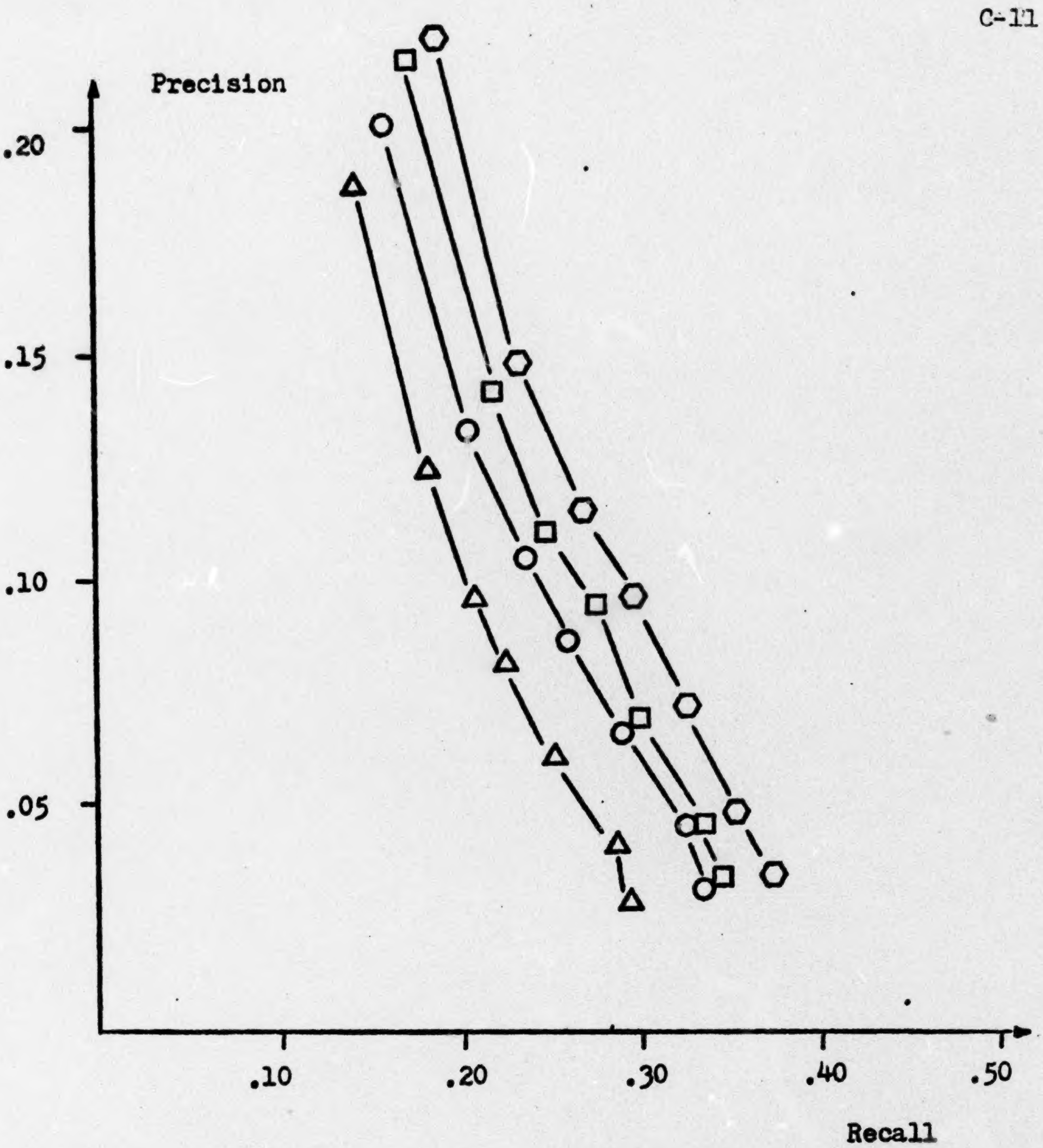
Figure C-5



Symbol	Profile	Rank
○	P_3	2½
⬡	P_3^*	1
□	$P_3^*(\delta = -1)$	2½
△	$P_1^*(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation
Hierarchy 2, Level 2

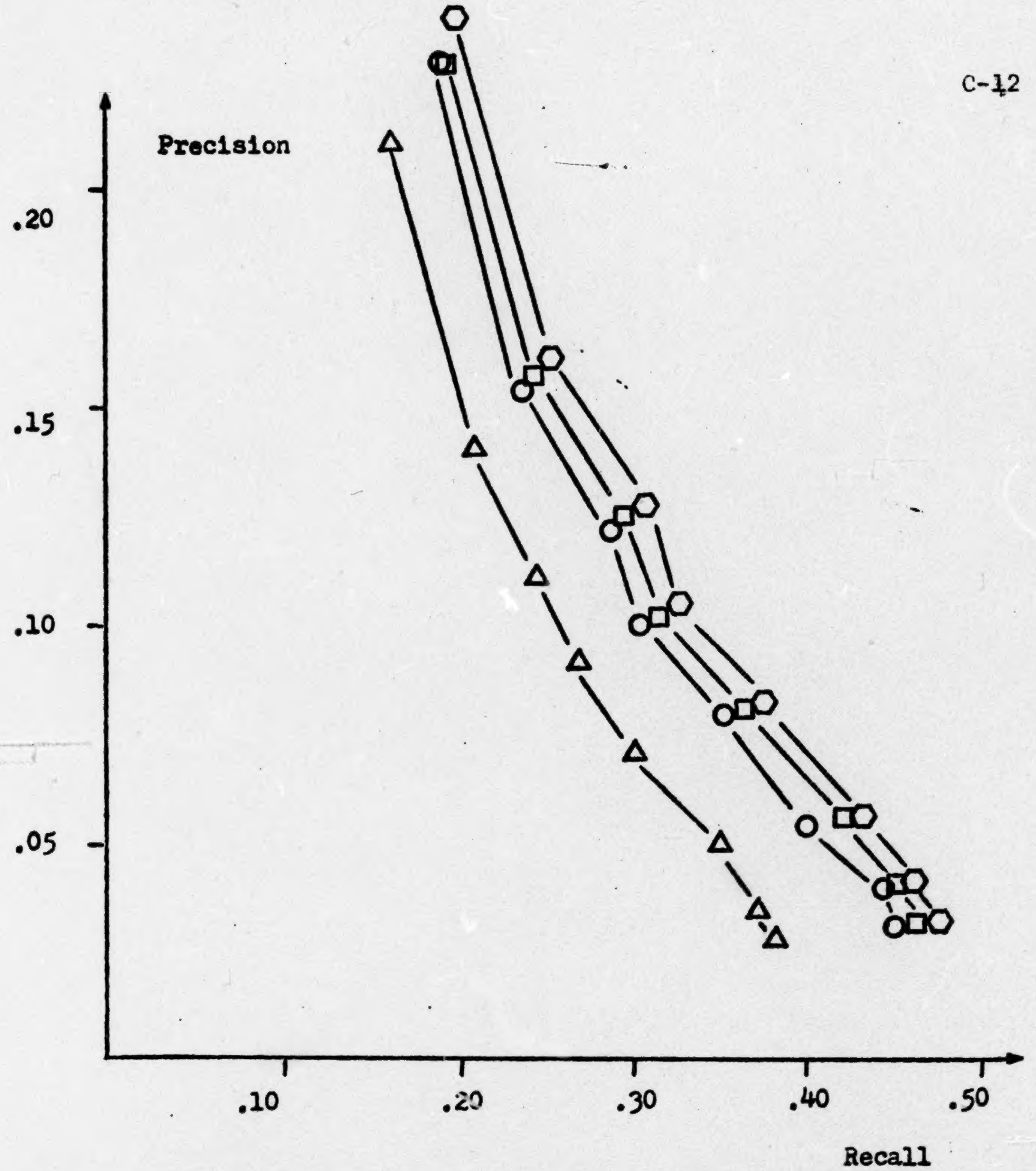
Figure C-6



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.635	.380	6	19.3	85	2
⬡	P^*_3	.660	.409	6	21.0	90	1
□	$P^*_3(\delta = -1)$	.650	.393	6	21.0	89	3
△	$P^*_1(\delta = -1)$	.617	.352	6	21.6	88	4

Confirmation Test Results--SMART Evaluation
Hierarchy 2, Narrow Search

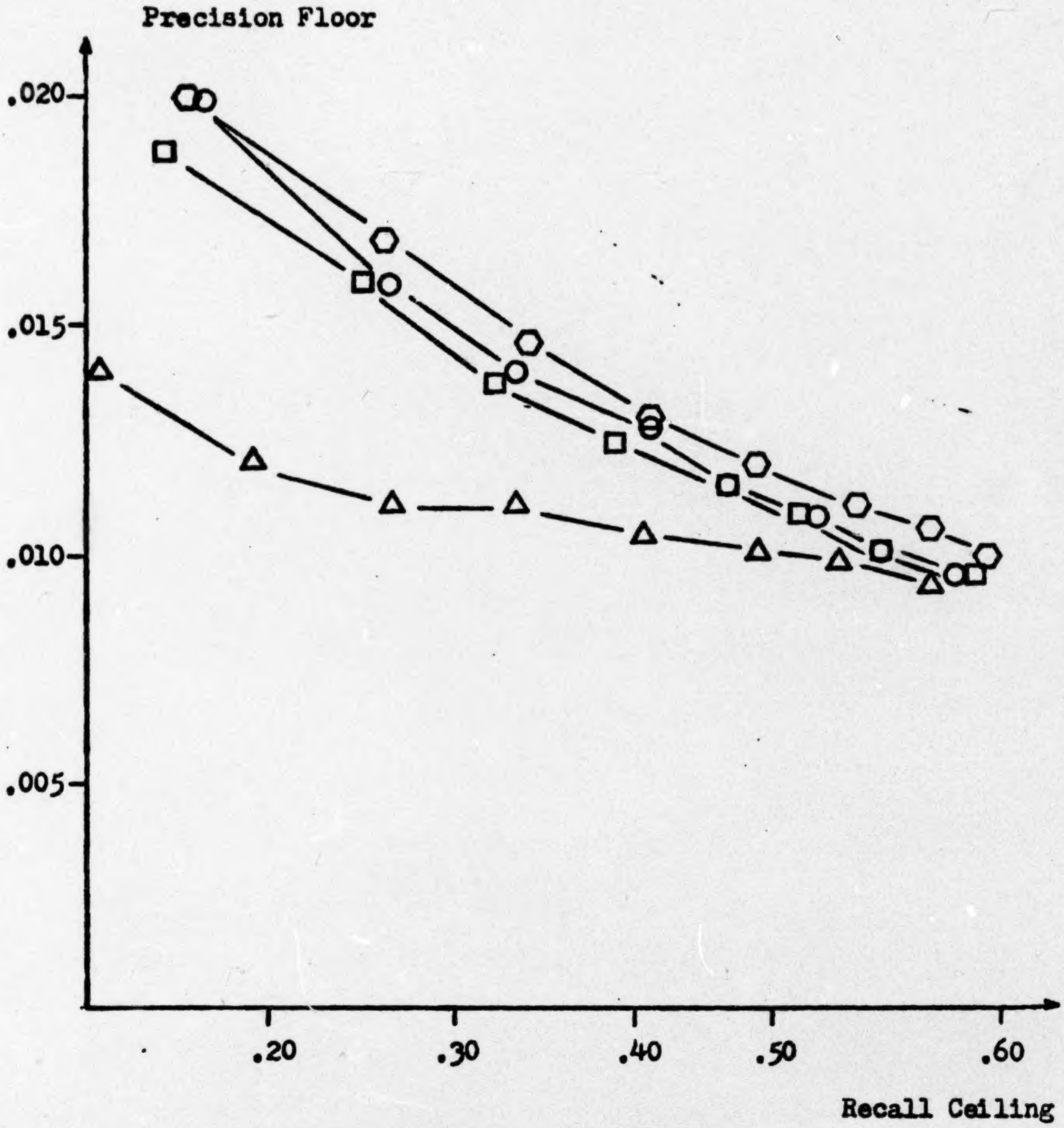
Figure C-7



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.693	.441	6	26.5	146	2
⬡	P^*_3	.707	.456	6	27.1	150	1
□	$P^*_3(\delta = -1)$	.695	.449	6	29.1	150	3
△	$P^*_1(\delta = -1)$	.643	.392	6	28.9	150	4

Confirmation Test Results--SMART Evaluation
Hierarchy 2, Broad Search

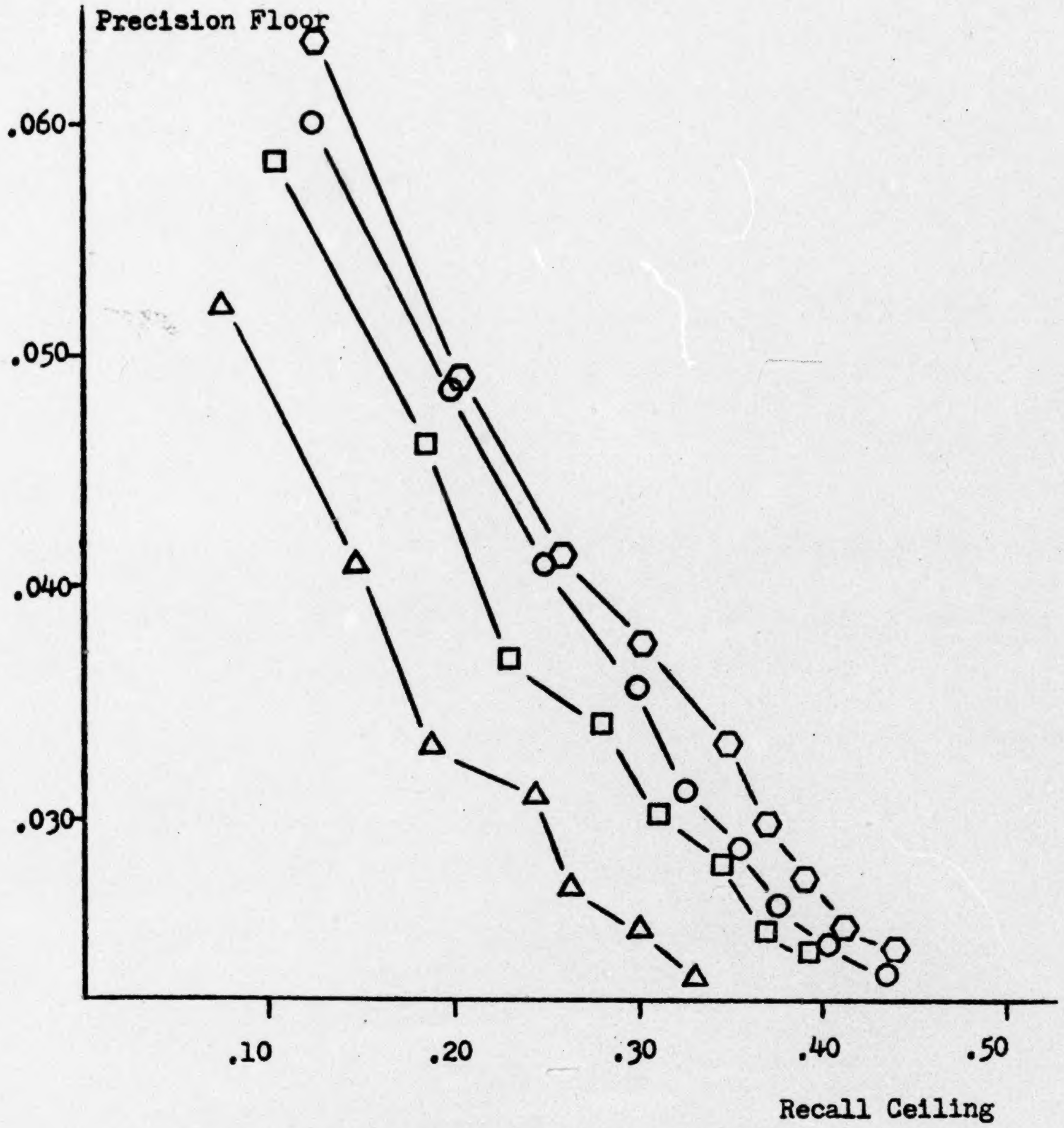
Figure C-8



Symbol	Profile	Rank
○	P_3	2½
⬡	P_3^*	1
□	$P_3^*(\delta = -1)$	2½
△	$P_1^*(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation
Hierarchy 3, Level 1

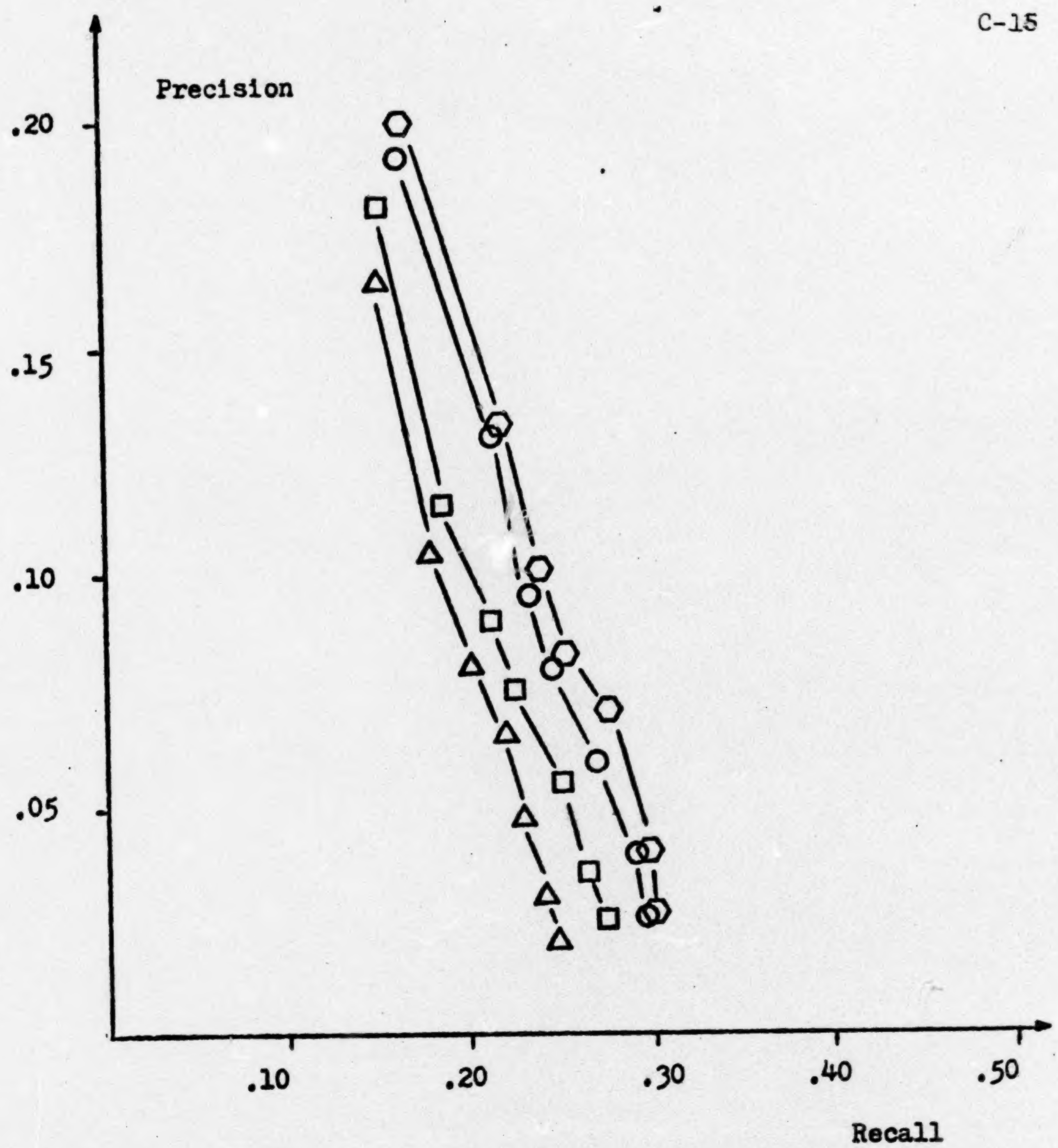
Figure C-9



Symbol	Profile	Rank
○	P_3	2
⬡	P^*_3	1
□	$P^*_3(\delta = -1)$	3
△	$P^*_1(\delta = -1)$	4

Confirmation Test Results--Cluster-oriented Evaluation
Hierarchy 3, Level 2

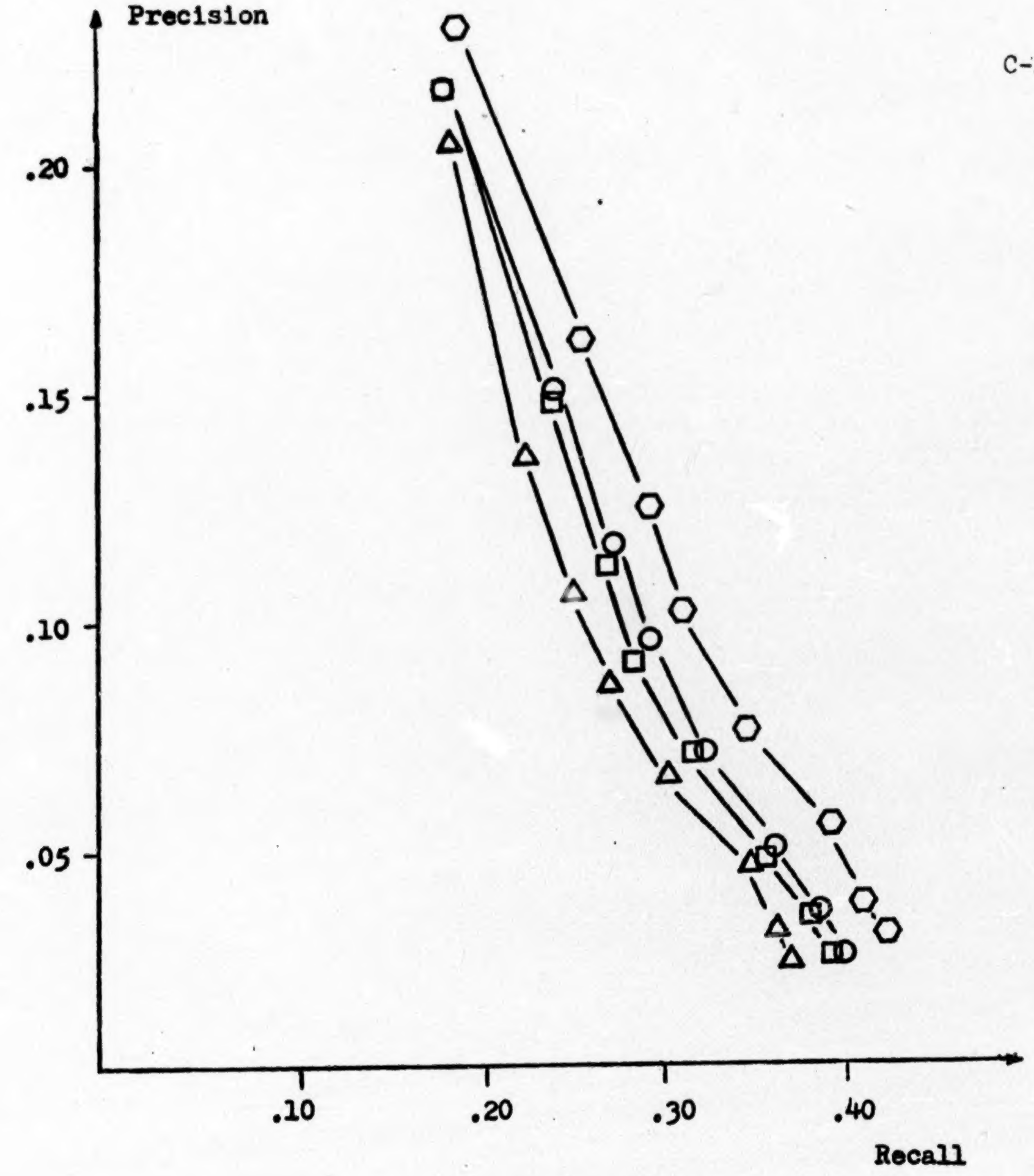
Figure C-10



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.615	.357	28	15.7	79	1½
⬡	P^*_3	.613	.361	28	16.1	81	1½
□	$P^*_3(\delta = -1)$	.609	.342	28	16.2	80	3
△	$P^*_1(\delta = -1)$	.577	.320	28	16.2	79	4

Confirmation Test Results--SMART Evaluation
Hierarchy 3, Narrow Search

Figure C-11



Symbol	Profile	NR	NP	C(1)	C(2)	C(3)	Rank
○	P_3	.659	.412	28	25.9	153	2½
⬡	P_3^*	.674	.430	28	26.0	155	1
□	$P_3^*(\delta = -1)$	.652	.405	28	26.3	153	2½
△	$P_1^*(\delta = -1)$	.639	.392	28	26.9	154	4

Confirmation Test Results--SMART Evaluation
Hierarchy 3, Broad Search

Figure C-12